Markt&Technik

Das Buch zum erfolgreichen 64'er-Einsteigerkurs: Henning packt aus

Henning Withöft
Andrew Draheim

64'er
Großer Einsteiger-Kurs

Leichtverständliche Einführung in die Welt des C64: Monitor ★ Diskettenstation ★ Basic-Programmierung: Grafik ★ Spiele ★ Musik

Auf 5¼"-Diskette im Format 1541/70/71 enthalten: Alle großen Beispielprogramme.

64'er - Großer Einsteiger-Kurs

64'er

Großer Einsteiger-Kurs

Leichtverständliche Einführung in die Welt des C64: Monitor ★ Diskettenstation ★ Basic-Programmierung: Grafik ★ Spiele ★ Musik

Henning Withöft
Andrew Draheim

Markt&Technik Verlag AG

CIP-Titelaufnahme der Deutschen Bibliothek

Withöft, Henning:
64'er [Vierundsechziger], großer Einsteigerkurs : leichtverständl. Einf. in d. Welt d. C64 :
Monitor, Diskettenstation, Basic-Programmierung ;
Grafik, Spiele, Musik / Henning Withöft ; Andrew Draheim. -
Haar bei München : Markt-u.-Technik-Verl., 1988.
ISBN 3-89090-668-0
NE: Draheim, Andrew:

15 14 13 12 11 10 9
91

ISBN 3-89090-668-0

Einbandgestaltung: Grafikdesign Heinz Rauner
Druck: Schoder, Gersthofen
Printed in Germany

Inhaltsverzeichnis

Einsteigerteil

Fortgeschrittenenteil

Vorwort

Personenkult um einen Computer oder Computerkult um eine Person – so dürfen wir den auspackenden Henning nicht verstehen. Die ganze Geschichte fing mit völlig anderem Ziel an.

Während meiner Zeit als freier Mitarbeiter für die Aachener Volkszeitung lernte ich Henning kennen, der es verstand, die verschiedensten Grimassen zu schneiden. Seltsam daran war, daß wir jahrelang auf dieselbe Schule gingen. Henning behauptet heute noch, er wäre in dieser Zeit sogar mal bei mir zu Hause gewesen, woran ich mich nun gar nicht mehr erinnern kann.

Irgendwann bei einem Bier in der Düsseldorfer Altstadt erzählte Henning, er würde auch gerne journalistisch tätig sein. Da gerade in Düsseldorf die Bundesgartenschau auf vollen Touren lief, riet ich: »O.K., schreib mal was über die BuGa und ich gebe es dann weiter.« Nun, die Anfangserfolge waren spärlich.

Kurze Zeit später fing ich beim 64'er-Magazin des Markt&Technik-Verlages als Redakteur an. Nun galt es, einen eigenen Autorenstamm aufzubauen. Für die Rubrik »Bücher« fiel mir Henning ein. Fachbücher konnte ich ihn zwar nicht testen lassen, denn er hatte damals von Computern noch keinen blassen Schimmer, doch literarische Bücher, die etwas mit Computern zu tun haben, konnte er bearbeiten. Ich schrieb ihm einen Merkzettel, mit Tips und Tricks zum Schreiben, und siehe da, sein erster Artikel war gut gelungen. »Vielleicht ist er doch ausbaufähig«, dachte ich mir und weitete unsere Zusammenarbeit aus.

Ich versuchte, Henning so viele Artikel zu geben wie möglich. Leider war die Auswahl aufgrund seines mangelnden Fachwissens begrenzt. Und genau dieses machte ihn zum geeigneten Autoren für einen Einsteigerkurs. Auf einer unserer zahlreichen Heftkritiken stellten wir Redakteure des 64'er-Magazins einstimmig fest, daß unsere Artikel für Einsteiger zu schwierig seien. Wir kamen auf die Idee, einen guten Journalisten einen

C 64 kaufen zu lassen und seinen ersten Umgang mit dem Computer als Erfahrungsbericht zu veröffentlichen.

Noch am selben Tag holte ich Henning per Telefon aus dem Bett (er war krank) und sagte: »Hast Du Lust einen Kurs zu schreiben, jeden Monat drei Seiten?« Er stimmte nach kurzem Zögern zu. Der Name »Henning packt aus« für die Serie entstand mehr oder weniger zufällig. Auf unserem Redaktionsplan mußte irgendein Name eingetragen werden. Ganz spontan entstand so »Henning packt aus«.

Mit dem Erfolg der neuen Serie hatte niemand so richtig gerechnet. Doch eine Woche nach der ersten Ausgabe kamen täglich Leserbriefe in die Redaktion. Einsteiger teilten mir mit Freude mit, daß es endlich einen Kurs gebe, den auch sie verständen; Fortgeschrittene und Profis schrieben, sie hätten solch einen Kurs bei ihren Startversuchen auch gerne gehabt. Selbst wenn Henning inhaltliche Fehler in seinen Serien hatte, reagierte niemand böse. Statt dessen boten seine Leser Hilfe an, schlugen einen Software-Tausch vor oder schickten Verbesserungen. Sie berichteten Henning von ihren Problemen mit dem Computer und mit der Familie. Henning war und ist für alle Einsteiger der Kumpel, der Leidensgenosse, der sich durch eine harte Materie durchbeißt. Es gab sogar Leser, die mit ihm Karate trainieren wollten – eines seiner Hobbys.

Henning und ich haben alle diese Leserbriefe gelesen und Kritik und Lob verarbeitet. Häufig fragten die Leser nach einem Buch zur Serie. Obwohl der Kurs »Henning packt aus« immer so ausgelegt war, daß der Einsteiger jederzeit ohne große Probleme einsteigen konnte, war es bei manchen Folgen nicht so ohne weiteres möglich, da ein Minimum an Grundwissen Voraussetzung war. So kamen wir auf die Idee, ein Buch zu schreiben, das umfassend in die Welt des C 64 einführt. Bereits erschienene Kursteile wurden überarbeitet, ausgeweitet und eine Reihe wichtiger Kapitel hinzugeschrieben. So entstand eine richtig runde Sache, die nicht nur dem C 64-Einsteiger einen guten Überblick gibt. Das Buch bildet den Grundpfeiler eines fundierten Computer-Wissens.

Henning und ich wünschen Euch, liebe Leser, viel Spaß und Erfolg mit dem C 64. Wir würden uns freuen, von Eurer weiteren Entwicklung zu hören. Vielleicht macht Ihr sogar aus Eurem Hobby später mal einen Beruf? Wir lassen uns überraschen.

Andrew Draheim

Vom Umgang mit diesem Buch

Dieses Buch enthält alle Informationen, die Ihr als Einsteiger für Eure weitere Arbeit mit dem C 64 braucht. Und doch sind sie nicht in geballter Form vorhanden. Beim Durcharbeiten werdet Ihr merken, daß Henning sich entwickelt. Sein Wissen ist im ersten Kapitel gleich Null, steigt aber stetig an. Wahrscheinlich wird es für Euch nicht zu unterscheiden sein, ob Ihr Euch mit Henning entwickelt oder Henning mit Euch. Tatsächlich ist beides der Fall. Öfter werdet Ihr Sprüche lesen wie: »Jetzt ist der Groschen gefallen« oder »da liegt der Hase im Pfeffer«.

Vielleicht lernt Ihr schneller als Henning. Dann überspringt einfach ein Kapitel. Dieses Buch ist so ausgerichtet, daß Ihr beliebig in den Kapiteln herumspringen könnt.

Hier sind wir beim Kernpunkt. Das Buch ist in zwei Teile gegliedert, einer für Einsteiger und einer für Fortgeschrittene. Nur wenige Kapitel im Einsteigerteil benötigen ein Vorwissen. Das heißt, theoretisch könnt Ihr auch mit Kapitel 3 dieses Buches anfangen, oder mit Kapitel 4, und später erst die vorherigen Kapitel durcharbeiten. So müßt Ihr Euch nicht durch Thematiken durchbeißen, die Ihr schon kennt.

Der Fortgeschrittenenteil baut vollständig auf den Einsteigerteil auf, d.h., dort könnt Ihr ebenfalls Kapitel überspringen. Vorausgesetzt wird jedoch das Wissen des ersten Teils. Diese Zusammenstellung macht dieses Buch auch als Nachschlagewerk nutzbar. Am Ende jedes Kapitels haben wir ein Resumee gezogen unter dem Titel »Das haben wir gelernt«. Oft helfen die stichpunktartigen Definitionen schon weiter.

Nichts wie ran an den Speck! Spaß beim Lesen und Computern wünschen Euch

die Autoren

Einsteigerteil

Kapitel 1

Kabel, Computer und ein wenig Programmieren

Ich habe es getan! Nach langem Zögern und vielen Plänen habe ich entschlossen zugeschlagen.

Ich ging in einen Computer-Shop und kaufte mit dem zusammengesparten Geld einen Commodore 64 und dazu ein Disketten-Laufwerk. Mit meinem Hinterwäldlertum ist jetzt Schluß, von nun an rede ich mit!

Alles steht ausgepackt vor mir auf dem Schreibtisch: der Computer, die schachtelförmige Disketten-Station, ein Gewirr von Kabeln und das Netzgerät. Mein ehrwürdiger Schreibtisch, der bisher mit einfachen Dingen wie Vokabeln oder Mathe-Hausaufgaben bedeckt war, bietet einen völlig neuen Anblick.

Voller Begeisterung mache ich mich an den Aufbau, das kann ja wohl nicht so schwer sein. Das Netzgerät: auf der einen Seite ein normaler Steckdosenstecker, auf der anderen Seite ein pfenniggroßer Anschluß für den Computer. Ich begebe mich auf die Suche, wo könnte er hineinpassen?

Aus dem Gehäuse des Computers gähnen mir diverse Löcher und Schlitze entgegen. In Gedanken sehe ich meinen schönen Computer verschmoren, weil ich den falschen Netzanschluß verwendet habe. Auf der rechten Seite, direkt neben dem Einschaltknopf, entdecke ich den richtigen (Bild 1.1).

Vorsichtig drücke ich den Stecker hinein und lege den Power-Knopf um. Nichts passiert, nur eine rote Kontrollbirne nimmt ihre Arbeit auf.

Bild 1.1: Die Anschlüsse auf der rechten Seite des Computers. Neben dem Einschaltknopf befindet sich der Netzanschluß.

Ohne Fernseher geht's nicht

Plötzlich fällt mir das Antennenkabel ein, ich muß erst den Fernseher anschließen! Das besagte Kabel paßt mit dem einen Ende in den Antenneneingang des Fernsehers. Das andere Ende paßt auf einen Anschluß an der Rückwand des Computers. Jetzt nur noch einschalten und ich sehe mich schon als Computerfachmann.

Doch außer der roten Kontrollbirne und einem rauschenden Bildschirm tut sich nichts. Kein Bild, kein Ton. Was soll ich tun? In diesem verzweifelten Moment kommt mir eine Idee. Bevor ich zum Telefon renne und um Hilfe bettele, erst einmal nachdenken. Ich habe den Computer mit dem Fernseher verbunden, bekomme aber kein Bild. Der Computer ist mein »Sender«, der Fernseher kann ihn aber nicht empfangen! Vielleicht muß ich den Fernseher auf den neuen Sender einstellen?

Ich wähle eine freie Programmtaste, zum Beispiel Programm 16 (jede andere tut es genauso) und drehe so lange am Programmsuchknopf, bis der Computer mich mit einem blauen Bildschirm, zwei Kopfzeilen und dem Wort »READY« empfängt. Ich habe es geschafft (Bild 1.2).

Auf dem Bildschirm sehe ich einen hellblauen Rand und ein dunkelblaues großes Rechteck. In diesem Rechteck steht »READY«, darunter blinkt ein weißes Quadrat. Dieses Quadrat nennt sich »Cursor« und zeigt meine Position auf dem Bildschirm an. Durch Drücken einzelner Tasten gelingt es mir, wunderschöne Zeichen zu produzieren. Der Cursor rückt bei jedem Buchstaben eine Position weiter.

Bild 1.2: Nach dem Einstellen des Fernsehers erscheint die Einschaltmeldung des C 64, er ist startbereit.

Weiter geht es mit der Einstellung der Bildschirmfarben. Da gibt es eine sogenannte Kontrolltaste namens [CTRL] (auf der Taste steht »CONTROL«). Sie spielt in dem Zusammenhang mit den Zifferntasten eine große Rolle. Ich drücke nacheinander die CTRL-Taste und dann die 9-Taste und soll dadurch in einen anderen Modus kommen. Es bleibt mir völlig schleierhaft, was unter Modus zu verstehen ist.

Auf jeden Fall sollen nach diesen beiden Tasten alle eingegebenen Zeichen revers erscheinen. Nicht mehr weiß auf blauem Hintergrund, sondern blau auf weißem Grund.

Weiß auf Blau und Blau auf Weiß

Trübsal überkommt mich. Auch nach dem zwanzigsten Zeichen bleibt alles beim alten. Bin ich doch zu dumm? Die vermeintliche Dummheit ist schnell geklärt. Die CTRL-Taste wird immer gleichzeitig mit einer anderen gedrückt (am besten wählt Ihr bei gedrückter CTRL-Taste die gewünschte Zifferntaste)! Ein neuer Versuch führt zum Erfolg. [CTRL] und [9] gedrückt, lassen die eingetippten Buchstaben revers erscheinen. Jetzt probiere ich es mit der Leer-Taste.

Sie ist jener Tasten-Riese am unteren Rand der Tastatur, auch Space-Taste genannt. Ein immer länger werdender heller Balken erscheint auf dem Bildschirm.

Die Bildschirmfarben werden mit Hilfe der CTRL-Taste und der Zifferntasten eingestellt. Ich drücke gleichzeitig [CTRL] und [8]. Dann lasse ich los und drücke eine Weile auf [SPACE]. Ein gelber Balken erscheint. Mit den anderen Zahlen entstehen verschiedene Farben, es funktioniert (Bild 1.3).

Bild 1.3: Mit Hilfe der CTRL-Taste und der Zifferntasten entstehen die tollsten Farbbalken.

Schreibmaschine mit Bildschirm

Die Tastatur zeigt für mich zuerst eine verwirrende Vielfalt. Viele Tasten sind mit mehreren Zeichen markiert. Die Tastatur sieht so aus, wie die einer Schreibmaschine, also wird sie auch so funktionieren. Ich versuche es mit den Worten »Ab die Post«. Schon beim ersten Buchstaben stimmt etwas nicht! Auf meinem Bildschirm befinden sich noch die bunten Balken von der Farbeinstellung und die revers geschriebenen Buchstaben. Direkt an den letzten gelben Balken schreibe ich in reverser, gelber Schrift »Ab die Post«. Was habe ich jetzt schon wieder falsch gemacht?

Auch heftigstes Fluchen und Ausprobieren der anderen Tasten zeigt keine Wirkung. Ich muß mein Gehirn einschalten! Da war doch vorhin dieser unverständliche Begriff »Modus«. Ich muß den Computer wieder in seinen »Normalzustand« zurückversetzen. Ich erhalte einen »sauberen Bildschirm« durch gleichzeitiges Drücken der SHIFT- und CLR/HOME-Taste. Jetzt stört nur noch der gelbe Cursor. Der verschwindet durch Drücken der Commodore-Taste ganz links unten in der Ecke und [7]. Wieder einmal schreibe ich »Ab die Post«, immer noch strahlen mich revers geschriebene Buchstaben

an. Durch Betätigen der CTRL- und 0-Taste verschwindet auch dieses Übel. Mein Satz steht in großen Buchstaben auf dem Bildschirm. Jetzt dämmert mir auch, was Modus heißt.

Modus ein, Modus aus

Allgemein bedeutet »Modus« Eigenschaft oder Zustand. Wir können also mit einem kleinen Handgriff unsere Tastatur in verschiedene Zustände bringen. Das geschieht mit [CTRL] in Zusammenhang mit einer weiteren Taste. [CTRL] allein hat keine Wirkung. Nach dem Einschalten ist der C 64 im Großschrift-/Grafik-Modus. In diesem Zustand kann ich große Buchstaben schreiben. Ich erreiche auch alle Grafikzeichen, davon aber später mehr.

Den reversen Modus habe ich am Anfang durch die Tasten [CTRL] [9] erreicht. Die Schreibweise [CTRL] [9] bedeutet: Die Tasten [CTRL] und [9] gleichzeitig drücken. Auf diese Weise erscheinen alle Buchstaben und der Hintergrund in umgekehrter Farbe. Diesen Modus kann ich durch [CTRL] [0] wieder verlassen. Ein weiterer Modus ist der Groß-/Kleinschrift-Modus, mit dem ich wie mit einer Schreibmaschine auf den Bildschirm schreiben kann. Ihn erhalte ich durch die Commodore-Taste links unten und die SHIFT-Taste. Der Rückweg ist der gleiche: [C=] [SHIFT] ergibt wieder den Großschrift- und Grafik-Modus.

Alles klaro? Dann weiter.

Der Kopf brennt

Wie schon gesagt, läßt sich die Tastatur in verschiedene Zustände schalten. Das heißt, daß dieselbe Taste, in verschiedenen Modi, verschiedene Zeichen ausdruckt. Ich schaue mir die Tastatur etwas genauer an. Dieses Unternehmen läßt meinen Schädel brummen. Einige Tasten haben Doppelfunktionen, wieder andere ergeben erst im Verbund die richtige Funktion.

Damit ich in den verschiedenen Ausdrücken nicht ertrinke, erstelle ich mir eine Funktionstabelle. Das Finden der Tasten bereitet keine Schwierigkeit, da sie alle beschriftet sind (Tabelle 1.1).

<SPACE>	Cursor rückt eine Position weiter, Freizeichen
<CTRL>	(CTRL leitet sich von ConTRoL ab.) Schaltet die verschiedenen Modi ein, zum Beispiel wie beschrieben von den Farbbalken auf normale Buchstaben, die CRTL-Taste wird immer nur gleichzeitig mit anderen Tasten verwendet!
<SHIFT>	Groß- und Kleinschreibung (nur im Groß-/Kleinschrift-Modus); bei Tasten mit mehreren Funktionen (zum Beispiel CLEAR/HOME) wird durch die SHIFT-Taste die obere angesprochen; beide SHIFT-Tasten haben die gleiche Funktion.
<RETURN>	Die auf dem Bildschirm sichtbare Information wird eingespeichert, das heißt, daß der Computer erst jetzt erfährt, was auf dem Bildschirm dargestellt ist.
<CLEAR/HOME>	Bei Betätigung dieser Taste springt der Cursor in die linke obere Ecke des Bildschirms (HOME-POSITION), SHIFT- und CLEAR/HOME-Taste zusammen löschen zusätzlich den gesamten Bildschirm.
<RESTORE>	(leitet sich von RESTORE ab = engl. »wiederherstellen«). Durch Drücken von RESTORE und RUN/STOP wird der Computer in den Ausgangszustand zurückgebracht.
<RUN/STOP>	RUN/STOP unterbricht den Ablauf eines Basic-Programms, der Computer stoppt einen Vorgang, RUN/STOP zusammen mit SHIFT lädt ein Programm vom Band und startet es.
<CRSR>	Das sind die beiden Tasten für jenes blinkende Quadrat, den Cursor; die linke verschiebt den Cursor nach unten (mit SHIFT nach links)
<INST/DEL>	löscht das vor dem Cursor stehende Zeichen, SHIFT zusammen mit INST/DEL erlaubt das Einsetzen von Zeichen in schon geschriebene Zeilen.
<Commodore>	besitzt mehrere Funktionen alleine: Grafikzeichen auf der linken vorderen Seite der Tasten werden erreicht. Commodore mit Zahlen: Umschalten auf andere Farben. Commodore mit SHIFT: Umschalten Groß-/Kleinschrift-Modus zu Großschrift/Grafik-Modus, und umgekehrt.

Tabelle 1.1: Die Tasten und ihre Funktionen auf einen Blick.

Nach der Theorie nun die Praxis. Die Tasten haben ein wenig von ihrem geheimnisvollen Wesen verloren, alle haben zumindest auf dem Papier eine bestimmte Funktion. Einige Tasten haben bei meinem momentanen Kenntnisstand keine sichtbare Wirkung auf dem Bildschirm. Doch plötzlich stoße ich auf ein Problem. Der Cursor, jenes Quadrat, das meinen Standort auf dem Bildschirm angibt, arbeitet nicht richtig.

Unten rechts auf der Tastatur befinden sich zwei Tasten, die zur Cursor-Steuerung vorgesehen sind. Auf einer steht »CRSR« und zwei Pfeile, die nach oben und unten zeigen. Die andere hat dieselbe Aufschrift, jedoch ein Pfeil zeigt nach links und einer nach rechts. Mit diesen beiden Tasten läßt sich der Cursor in alle vier Himmelsrichtungen über den Bildschirm bewegen. Werden sie alleine gedrückt, bewegt er sich nach rechts und nach unten; in Verbindung mit [SHIFT] nach links oder nach oben.

Der Cursor bereitet mir im Moment einige Probleme: Was ich auch tue, er wandert immer nach oben oder nach links. Er funktioniert nicht richtig und läßt sich nicht nach rechts bewegen! Also doch ein defekter Computer, ich habe es geahnt.

Bloß keinen Ärger

Mißtrauisch beäuge ich den Computer und überlege, ob er wohl Streit anfangen will. Da fällt mir etwas auf. Eine Taste am linken Rand der Tastatur ist offenbar gedrückt. Sie ist mit »SHIFT/LOCK« beschriftet. Bin ich vorhin bei meinem Herumtippen an diese Taste gekommen, oder saß sie schon immer so tief? Ich probiere sie aus. Siehe da, jetzt ist sie auf gleicher Höhe mit den anderen. Das war es! Es gelingt mir nun, den Cursor nach unten oder rechts zu bewegen. Die SHIFT/LOCK-Taste hat eine feststellende Funktion. Bei gedrückter Taste arbeitet der Computer, als sei die SHIFT-Taste ständig aktiv. Aus diesem Grunde konnte der Cursor auch nur nach oben und nach links bewegt werden.

Programmieren kein Problem!

Wir schreiben ein kleines Programm! Ich probiere einen Befehl namens »PRINT« aus. Er gibt dem C 64 den Auftrag, etwas auf den Bildschirm zu schreiben. Ich gebe ein:

```
PRINT HENNING LEGT LOS
```

und drücke [RETURN]. Der Computer schreibt eine 0 und »READY«. Eigentlich sollte er jetzt »HENNING LEGT LOS« schreiben. Ich habe die Anführungszeichen vergessen!

```
PRINT "HENNING LEGT LOS"     <RETURN>
```

Alles läuft primstens. Der Befehl PRINT bedeutet soviel wie »schreibe«, die Anführungszeichen beschränken die zu schreibenden Zeichen. Alle Worte, die auf den Bildschirm geschrieben werden sollen, müssen in Anführungszeichen gesetzt werden, sonst

tut der C 64 gar nichts! Der Umgang mit den Anführungszeichen ist manchmal etwas schwierig. Es kann zum Beispiel passieren, daß das Schließen der Anführungszeichen vergessen wird. In diesem Falle erhaltet Ihr völlig unbekannte Symbole auf dem Bildschirm. In diesem Fall gibt es nur eines: Die Anführungszeichen schließen und das nie wieder vergessen!

Mit [SHIFT] [CLR/HOME] lösche ich den Bildschirm, dann leere ich den Programmspeicher mit dem Wort NEW und [RETURN]. Ich tippe ein:

```
10 PRINT "HENNING LRGT LOS"   <RETURN>
20 GOTO 10              <RETURN>
```

Das eigentliche Programm steht auf dem Bildschirm. Es muß nur noch gestartet werden. In diesem Moment fällt mir auf, daß ich »LRGT« statt »LEGT« geschrieben habe. Schnell ist meine Tastatur-Tabelle zur Hand.

Ich gehe mit dem Cursor auf das »G« nach dem falschen »R« und drücke [INST/DEL]. Das »R« verschwindet. [SHIFT] [INST/DEL] verschafft mir Raum, ich setze das fehlende »E« ein. Das Programm startet, nachdem ich RUN eingetippt und die RETURN-Taste gedrückt habe. Mein erstes Programm läuft und schreibt superschnell immer wieder »HENNING LEGT LOS«. Was bedeuten nun die einzelnen Befehle?

Da sind zunächst die Nummern vor den beiden Zeilen. Sie bestimmen, welche Befehle der C 64 zuerst ausführen soll. Die Zeile mit der niedrigsten Nummer nimmt er sich als erstes vor, die mit der höchsten als letztes. Theoretisch könnte vor der ersten Zeile auch eine »1« und vor der zweiten eine »2« stehen, also:

```
1 PRINT"HENNING LEGT LOS"
2 GOTO 1
```

oder aber:

```
1 PRINT"HENNING LEGT LOS"
5 GOTO 1
```

Es hat sich in der Praxis jedoch bewährt, Zeilennummern in Zehnerschritten zu nehmen. Auf diese Weise können nachträglich Zeilen eingefügt werden, zum Beispiel:

```
10 PRINT"HENNING LEGT LOS"
15 PRINT"VOLLE KANNE!"
20 GOTO 10
```

Bei Einerschritten ist das nicht mehr so ohne weiteres möglich.

Das ist unser erstes Programm! Ein Programm bedeutet eine Folge von Befehlen, die numeriert und mit RUN gestartet werden.

In unserem ersten Programm befiehlt Zeile 10 dem Computer »HENNING LEGT LOS« zu schreiben. Nachdem dieser PRINT-Befehl ausgeführt ist, folgt Zeile 20. Diese besagt, daß der Computer wieder in die Zeile 10 gehen und den dortigen Befehl ausführen soll (GOTO 10). Diese beiden Programmzeilen bilden eine Endlosschleife, in der der Computer immer wieder »HENNING LEGT LOS« untereinander auf den Bildschirm schreibt. Nach kurzer Zeit scheint das Programm stillzustehen. Auf dem Bildschirm steht in jeder Zeile »HENNING LEGT LOS«, nur die unterste flackert ein wenig. Unser Programm läuft trotzdem weiter! Das Programm schreibt in jede Zeile des Bildschirms den Inhalt der Anführungszeichen. Wenn das Programm den unteren Rand des Bildschirms erreicht, schreibt der C 64 den Satz aus Zeile 10 immer wieder in die letzte Bildschirmzeile und schiebt alle anderen nach oben. Das geht so schnell, daß wir glauben, der Bildschirm steht still (ich stoppe das Programm durch die RUN/STOP-Taste).

Bei folgendem Programm seht Ihr es ganz deutlich:

```
10 PRINT"HENNING"
20 PRINT"LEGT LOS"
30 GOTO 10
```

Ein grafischer Rundgang

Ein paar kleine Zeichnungen tun nun not! Wir haben lang und breit über die verschiedenen Modi der Tastatur gesprochen. Ihr habt Euch bei dem Betrachten der Tastatur sicherlich gefragt, was die Zeichen auf der Vorderseite der Tasten zu bedeuten haben und wie man sie darstellen kann. Diese Zeichen sind Grafikzeichen. Alle Zeichen auf der linken Seite werden durch gleichzeitiges Drücken der Commodore-Taste und der jeweiligen Buchstabentaste erzeugt. Probiert das mal aus. Bei [C=] [R] erhaltet Ihr nach mehrfachem Drücken eine Art Zaun. Die anderen Buchstaben bergen so manche Überraschung. Alle Zeichen auf der rechten Seite der Tasten entstehen mit Hilfe der SHIFT-Taste. [SHIFT] [Q] produziert einen Ball. Die Buchstaben A, S, Z und X ergeben Pik, Herz, Karo und Kreuz, die vier Spielkartenfarben! Mit all diesen Hilfsmitteln können durch Einsatz der Cursor-Tasten die tollsten Grafiken gemalt werden. Wer hätte das gedacht: Schon am ersten Tag produzieren wir Computer-Grafiken! Aber es kommt noch toller. Blättert schnell zum nächsten Kapitel.

Das haben wir gelernt

Cursor: Ein kleines Quadrat, das die Position anzeigt, an der wir ein Zeichen oder einen Buchstaben setzen wollen.

Modus: Ein Zustand der Tastatur. Verschiedene Modi belegen die Tasten mit verschiedenen Zeichen.

CTRL-Taste: Mit dieser Taste können wir in Verbindung mit anderen Tasten unterschiedliche Modi erzeugen.

CRSR-Tasten: Mit ihnen steuern wir den Cursor.

PRINT: Eine Anweisung, die dem C 64 befiehlt, einen Text, der zwischen Anführungszeichen steht, auf den Bildschirm zu schreiben.

GOTO: Der C 64 springt innerhalb eines Programms in die Zeile mit der Nummer, die nach dem GOTO steht. Die Anweisung GOTO 20 läßt den C 64 in Zeile 20 springen und die darin stehenden Befehle ausführen.

Kapitel 2

Massenhaft Informationen

Nach einer gut durchschlafenen Nacht und einem reichlichen Frühstück setze ich mich wieder an den C 64. Bei eingeschaltetem Monitor und C 64 springt mir auch wieder die bekannte Einschaltmeldung ins Gesicht. Nichts von gestern ist übriggeblieben.

Was unser Gehirn kann, kann der C 64 noch lange nicht: eine Zeitlang ausschalten und anschließend mit alten Informationen weiterdenken. Voller Entsetzen stelle ich fest, daß mein Computer sein Gedächtnis verloren hat.

Doch der C 64 hat wie jeder Computer eine Eigenart, die unser Gehirn nicht besitzt – er kann sich schnell seine vergessenen Informationen wieder holen. Massenspeicher heißt die Lösung.

Bei diesem Wort liegt die Vermutung nahe, daß es sich dabei um Geräte handelt, die massenhaft Informationen aus dem Computer speichern können. Und diese Annahme ist richtig. Für den C 64 stehen uns zwei Arten von Speichermedien, wie Massenspeicher auch genannt werden, zur Verfügung: die Datasette und das Disketten-Laufwerk.

Daten auf Tonband

Die Datasette (Bild 2.1) erinnert stark an einen herkömmlichen Kassettenrecorder. Tatsächlich hat sie mit ihm sehr vieles gemeinsam. Der einzige Unterschied besteht darin, daß die Datasette keine Musik aufnimmt und wiedergibt, sondern Daten.

Ihr könnt das hören, wenn Ihr mal eine Datenkassette auf einem Kassettenrecorder abspielen laßt. Ein furchtbares Piepsen und Rauschen dringt aus den Lautsprechern der Stereo-Anlage. Für Eure Ohren ist das eine Qual, für den C 64 bestes Futter.

Bild 2.1: Die Datasette ähnelt einem Kassettenrecorder.

An der Datasette befindet sich ein Kabel mit einem flachen Stecker. Diesen müssen wir in den Eingang mit der Aufschrift »Cassette« stecken. Er befindet sich auf der Rückseite des C 64. Über diesen Eingang (Port) können der C 64 und die Datasette Daten und Informationen austauschen. Er versorgt die Datasette auch mit Strom. Wir können unser erstes Speichermedium direkt ausprobieren. Legt eine handelsübliche Kassette in die Datasette, spult bis zum Bandanfang vor und gebt folgendes Programm in den C 64 ein:

```
10 PRINT"HENNING PACKT AUS"
20 GOTO 10
```

Nun schreiben wir nicht wie gewohnt RUN, sondern:

```
SAVE"HENNING"
```

und drücken die Taste [RETURN]. SAVE ist der Befehl zur Datensicherung auf einen Massenspeicher. Dieser Befehl ist, wie alle Basic-Befehle, der englischen Sprache entnommen und bedeutet »sichern«. Das Wort in Anführungsstrichen hinter dem Befehl ist der Name, unter welchem wir unsere Informationen speichern beziehungsweise sichern wollen. Er kann bis zu 16 Buchstaben lang sein. Der C 64 meldet sich nun mit der Nachricht: »PRESS RECORD & PLAY ON TAPE«. Das ist englisch und bedeutet, daß wir die Aufnahme- und die Wiedergabe-Taste auf der Datasette gleichzeitig drücken sollen. Ist das getan, gerät das Band in Bewegung und die Aufnahme erfolgt. Der C 64 bestätigt dies mit: »SAVING HENNING«, was übersetzt »ich sichere HENNING« bedeutet. Wenn er fertig ist, macht er das mit einem »OK« und »READY« deutlich.

So, nun wollen wir doch mal sehen, ob das alles so klappt, wie wir uns das vorstellen. Band zurückspulen, Computer kurz aus- und dann wieder einschalten, der Test beginnt. RUN und [RETURN] zeigen, daß der Knabe wirklich nichts mehr behalten hat. Außer einer READY-Meldung tut sich nämlich nichts. Ich gebe ein:

```
LOAD"HENNING"
```

Der C 64 weist mich an, die Wiedergabe-Taste zu drücken. Auf den Monitor schreibt er: »PRESS PLAY ON TAPE«. Nachdem ich seinen Willen befolge, verschwinden Text und Cursor vom Bildschirm. Dann meldet er: »FOUND HENNING«. Das war es dann auch. War mein Name zuviel für ihn? Es gibt eine andere Erklärung: Der C 64 wartet darauf, daß ich die Taste mit dem Commodore-Zeichen [C=] drücke (links unten auf der Tastatur). Jetzt beginnt er zu laden, das vertraute READY erscheint und der Cursor befindet sich wieder auf dem Bildschirm. RUN und [RETURN] zeigen mir, daß der C 64 sein altes Gedächtnis wieder hat. Er schreibt auf den Monitor unentwegt »HENNING PACKT AUS«.

Die Datasette hat nur einen kleinen Teil des Kassettenbandes mit Daten beschrieben. Ich frage mich, ob nicht mehrere Programme auf einer Kassette gespeichert werden können. Ich versuche es gleich und tippe ein:

```
NEW
10 PRINT"HENNING HAUT REIN"
20 GOTO 10
```

und speichere mit SAVE"TEST" [RETURN] das neue Programm. Da ich eben das alte Programm geladen und die Kassette nicht zurückgespult hatte, sichert der C 64 das neue direkt dahinter. Nach dem Speichervorgang spule ich bis zum Anfang zurück und versuche mit LOAD"TEST" zu laden. Die Datasette läuft und läuft, bis die Meldung FOUND TEST erscheint. Nach Drücken von [C=] (Commodore-Taste) lädt der C 64 das Programm mit dem Namen »TEST«. Ich gebe RUN ein und auf dem Monitor wird »HENNING HAUT REIN« geschrieben. Es funktioniert also. Der C 64 ist in der Lage, Programme anhand von Namen zu unterscheiden!

Die Datasette eignet sich hervorragend zur Datensicherung mit dem C 64. Aber sie hat Nachteile, die einem bald auf den Wecker gehen. Es ist logisch, daß Ihr immer mehr Daten und Programme speichert, je länger Ihr mit dem C 64 arbeitet. Schnell hat man 30, 50 oder gar 100 Programme zusammen. Was für ein Haufen an Kassetten sammelt sich da an. Na klar, wie wir wissen, können ja mehrere Programme hintereinander auf einer Kassette gespeichert werden. Richtig gute und lange Progamme füllen jedoch oft eine oder gar zwei Kassettenseiten. Zudem werdet Ihr schnell merken, daß die ewige Warterei, bis ein Programm geladen ist, ganz schön nerven kann.

Zum Glück gibt es ein weiteres Speichermedium für den C 64, die Disketten-Station (Bild 2.2). Mit ihr können viele Daten und Programme auf einer Diskette (Bild 2.3) gesichert werden.

Bild 2.2: Die Disketten-Station erlaubt schnelleren Umgang mit Programmen als die Datasette.

Bild 2.3: Das Futter der Disketten-Station: eine Diskette.

Mit der Disketten-Station können wir alle Nachteile der Datasette umgehen. Jeder, der viel mit dem C 64 arbeitet, sollte sich daher ernsthaft überlegen, einen solchen Massenspeicher anzuschaffen. Sie ist längst Standard-Ausrüstung für den C 64. Mehr darüber im nächsten Kapitel, schlagt schnell um.

Das haben wir gelernt

Massenspeicher: Geräte, mit denen man eine Menge Daten außerhalb des Computers festhalten (speichern) kann, um sie später bei Bedarf aufrufen zu können.

LOAD: Mit diesem Befehl können Daten von einem Massenspeicher geladen werden. Unmittelbar hinter ihm folgt der Programm-Name in Anführungsstrichen, z.B.

```
LOAD"HENNING"
```

SAVE: Das Gegenstück zu LOAD. Mit diesem Befehl werden Daten, die sich im Computer befinden, auf einem Massenspeicher gesichert, z.B.

```
SAVE"HENNING"
```

Datasette: Ein Massenspeicher, der wie ein Kassettenrecorder ohne Lautsprecher aussieht. Für den Anfang eine gute Sache, auf Dauer aber unzureichend.

Disketten-Station: Ein Massenspeicher, der dem Standard entspricht. Sie ist zum Beispiel wesentlich schneller und übersichtlicher zu handhaben als eine Datasette.

Kapitel 3

So geht's mit der Disketten-Station

Mein Computer-Terminal vergrößert sich. Ich habe mir die Disketten-Station 1571 von Commodore angeschafft, ein weiterer Schritt zum Profi. Meine Arbeit am C 64 wird nun umfangreicher, dafür aber einfacher und schneller.

In der Verpackung befinden sich das Disketten-Laufwerk, ein Netz- und ein Übertragungskabel. Im Gegensatz zur Datasette wird die 1571 nicht vom Computer mit Strom versorgt. Sie hat ein eigenes Netzteil und muß extra angeschlossen werden. Das Übertragungskabel verbindet Disketten-Station und C 64. Es ist für den Datenaustausch da. Achtung Leute! Bevor wir die Disketten-Station an den C 64 anschließen, müssen die Geräte ausgeschaltet sein! Ansonsten besteht die Gefahr, daß es irgendwo in den Geräten Puff macht und der Spaß ein vorzeitiges Ende hat. Wenn wir dies beachten, bereitet der Anschluß keine Schwierigkeiten.

An der Rückseite des C 64 existiert ein Eingang mit der Aufschrift »SERIAL«. Das bedeutet soviel wie »in Reihe« oder »nacheinander« und hat etwas mit Datenübertragung zu tun. Aber das interessiert uns im Moment wenig. Später gehen wir genauer darauf ein. In diesen Eingang stecken wir das Übertragungskabel. Das andere Ende gehört in eine der beiden Buchsen an der Rückseite des Disketten-Laufwerkes. Es ist egal, welche Buchse wir uns aussuchen. Beide funktionieren auf die gleiche Weise. Die zweite Buchse ist für den Anschluß eines zweiten Disketten-Laufwerks vorgesehen. So, nachdem alles angeschlossen ist, kann's losgehen.

Ins Laufwerk gehört, wie der Name schon sagt, eine Diskette. Eine Diskette ist eine viereckige Platte, etwas größer als eine Compact-Disk. In dieser Platte befindet sich eine kleine Magnetscheibe aus flexiblem Material. Auf eine solche Diskette kann man wie auf einer Kassette Daten speichern und sie in kürzester Zeit wieder in den C 64 laden. Wie wir schon wissen, ist der C 64 ohne uns ziemlich dumm. Er benötigt Material, »Software« (Programme) genannt, mit dem er arbeiten kann. Dieses bekommt er von der Diskette. Spiele zum Beispiel können in den C 64 gefüttert werden. Das können wir sofort ausprobieren, denn wir alle besitzen eine solche Diskette. Sie liegt dem Buch bei.

Jetzt brauchen wir nur noch einen Joystick. Ihn stecken wir in den Joystick-Port 2 an der rechten Seite des C 64. Einen Joystick kennt eigentlich jeder, er ist eine Art Lenkung für das Spiel. Das Abenteuer kann beginnen.

Denkste! Plötzlich fällt mir auf, daß schon etwas im Schlitz der Disketten-Station steckt, denn eine weiße Lasche ragt aus ihr heraus. Hierbei handelt es sich um eine Transport-Sicherung. Es ist dicke Pappe in Form einer Diskette mit einer Lasche zum Rausziehen. Das Disketten-Laufwerk hat im Innern den Schreib-/Lesekopf, der Daten auf Diskette schreibt und auch von ihr liest. Er ist mit dem Tonarm eines Plattenspielers vergleichbar. Da er sehr empfindlich ist, muß er für einen Transport gesichert werden.

Das Laufwerk besitzt an der linken Seite einen Hebel, der in die Waagerechte gedreht werden muß. Nun kann die Transport-Sicherung herausgenommen werden. Ich schalte C 64, Fernseher und Disketten-Laufwerk ein. Die Diskette kann jetzt mit dem Aufkleber nach oben eingeschoben werden. Den Sicherungshebel wieder in die Senkrechte drehen und alles ist bereit. Der Befehl zum Laden unseres Spiels lautet:

```
LOAD"GIANA SISTERS",8,1
```

LOAD bedeutet Laden und der Name in den Anführungsstrichen definiert, welches Programm geladen wird. Der Unterschied zum Ladebefehl bei der Datasette ist das angehängte »,8«. Es ist ja denkbar, daß wir zwei oder gar drei Massenspeicher an den C 64 angeschlossen haben. Durch das Komma und eine folgende Zahl erkennt er, von welchem Gerät er laden soll. Das Disketten-Laufwerk hat die Gerätenummer 8, ein zweites Disketten-Laufwerk hätte die Nummer 9. Wollten wir demnach von diesem Laufwerk laden, hieße der Befehl:

```
LOAD"GIANA",9
```

Ich habe aber nur ein Disketten-Laufwerk und lade daher von Nummer 8. Nach der Befehlseingabe erscheint die Meldung »SEARCHING FOR GIANA SISTERS« auf dem Bildschirm, und etwas später »LOADING«. Jetzt müssen wir RUN eingeben; nun habt Ihr einen ersten Einblick, was wir später mit dem C 64 alles machen können. Auf dem Bildschirm erscheint eine farbige, zerfließende Diskette. Nach Druck auf die Leertaste seht Ihr eine Grafik, das Titelbild des Spiels »The Great Giana Sisters«. Wenn Ihr einen Moment wartet, hört Ihr Musik von Chris Hülsbeck, einem bekannten Musik-Programmierer.

Hier noch ein kleiner Hinweis zum Laden des Spiels. Zuerst muß der Joystick an den hinteren Eingang angschlossen werden. Nach Eingabe von RUN drücken wir zweimal die Leertaste. Der C 64 braucht jeweils einige Zeit, bis er das Programm geladen hat. Das heißt, er muß noch eine Datei mit dem Namen »1.PRG« nachladen. Laßt Euch dadurch aber nicht verunsichern, auch nicht von bunten Streifen auf dem Bildschirm.

»The Great Giana Sisters« ist ein beliebtes Spiel aus dem Software-Hause Rainbow Arts. Auf unserer Diskette befindet sich eine Demonstration. Ihr könnt zwei Spielebenen lang spielen, dann ist Ende oder Neustart angesagt. Das vollständige Spiel besitzt 99 Spielebenen (Level) und ist im Handel erhältlich. Die kleine Giana wird mit dem Joystick gesteuert. Ziel ist es, möglichst viele Diamanten einzusammeln. Giana kann gegen Wände springen und Tiere abwehren, nur in die Gräben darf sie nicht fallen.

Nun ist eine kleine Denkpause angesagt. Entspannt Euch ein wenig mit der kleinen Giana aus Milano und bewundert, was der C 64 alles kann.

Das wahre Gesicht

Die Disketten-Station 1571 von Commodore legt los. Bisher war sie brav und friedlich, jetzt zeigt sie, was wirklich in ihr steckt. Sie ist längst nicht so unscheinbar, wie sie aussieht! Ich habe genug gespielt und will weiter! Mit einem Satz Leer-Disketten versorgt, versuche ich neue Geheimnisse zu ergründen.

Die Disketten-Station hat mir bisher keine Probleme bereitet. Meine erste Attacke besteht darin, daß ich mein kleines Computerprogramm mit meiner Disketten-Station auf eine meiner neuen Disketten speichern will. SAVE"PROGRAMMNAME",8 ist der Speicherbefehl, mein Programm lautete:

```
10 PRINT "HENNING LEGT LOS"    <RETURN>
20 GOTO 10                     <RETURN>
```

(Zuerst das Programm eingeben, dann den Speicherbefehl.)

```
SAVE"HENNING",8
```

(Achtung, falls Ihr mal aus Gewohnheit »,8« vergeßt, hilft Drücken von RUN/STOP weiter.) SAVE ist wieder der Befehl zum Sichern eines Programms. »,8« sagt wie bei LOAD, daß auf Diskette gespeichert wird. Mit einem leisen Summen nimmt das Gerät seine Arbeit auf. Zufrieden lehne ich mich in meinen Sessel zurück und genieße die Freuden des Computer-Himmels, einfach genial!

Abgestürzt

Irgend etwas stimmt nicht! Auf dem Bildschirm steht zwar »SAVING HENNING«, aber die Disketten-Station hört gar nicht mehr auf zu arbeiten! Habe ich da vielleicht etwas kaputtgemacht, oder hat mein kleiner Bruder daran herumgefummelt? Was soll ich tun? Die Disketten-Station läuft und läuft, ohne daß sich etwas ändert. Mir bleibt nur eine Notlösung. Zu einem Zeitpunkt, an dem nur die rote Kontrollampe leuchtet, nehme ich die Diskette aus dem Laufwerk und schalte es ab.

Große Ratlosigkeit macht sich breit. Was habe ich falsch gemacht?

Langsam, nach zähem Ringen mit meinen grauen Zellen, zeichnet sich ein Silberstreif des Verstehens ab. Die neue Diskette konnte noch nicht richtig arbeiten!

Ich habe bei meinem Speicherversuch einen grundlegenden Fehler gemacht. Eine leere Diskette muß erst betriebsbereit gemacht werden.

Dieser Vorgang nennt sich »Formatieren«. Eine unformatierte Diskette ist wie eine Schallplatte ohne Tonrillen, nur das Material ist vorhanden.

Das Disketten-Laufwerk ist auf bestimmte »Markierungen« auf der Diskette angewiesen, ohne die es nicht festlegen kann, wo sich der Lesekopf (vergleichbar mit dem Tonarm eines Plattenspielers) gerade befindet.

Fabrikneue Disketten besitzen diese Markierungen noch nicht.

Meine Disketten-Station fand keine Markierung, deswegen hörte sie nicht zu rattern auf und ich mußte sie ausschalten.

Kreis im Viereck

Das Herzstück einer Diskette ist eine runde, beschichtete Kunststoffolie. Sie wird von einer Schutzhülle umschlossen (Bild 3.1), die Schmutz und Staub von der empfindlichen Magnetfolie fernhält. In dieser Schutzhülle befindet sich ein Schutzvlies.

Beim Betrieb in der Disketten-Station wird die Diskette gedreht. Falls sich nun doch kleinere Dreckpartikel auf der Magnetfolie befinden, werden sie vom Schutzvlies abgefangen. Auf diese Weise ist die Diskette recht gut geschützt.

Trotz allem bleibt sie Dreck oder Staub gegenüber sehr empfindlich!

Beim Formatieren einer Diskette gehen die bisher auf ihr gespeicherten Daten verloren. Alte Disketten mit unwichtigen Programmen können so wieder einsatzbereit gemacht werden.

Bild 3.1: Die »Innereien« einer Diskette. Die weiße Schicht verhindert ein Zerkratzen der drehenden Magnet-Scheibe.

Bei der Formatierung »prägt« die Disketten-Station »Spuren« und »Sektoren« auf die Folie. Spuren sind kreisförmig um das sichtbare Zentralloch angebrachte Ringe. Die Commodore-Disketten-Stationen prägen 35 solcher Ringe auf eine Diskette.

Diese Ringe sind nochmals in gleich große Blöcke unterteilt. Man nennt sie Sektoren, kleine übersichtliche Abschnitte, in denen sich der Schreib-/Lesekopf zurechtfinden kann.

OPEN 15,unit,15,"N(drive):name,id" ist der überaus einleuchtende Befehl für die Formatierung einer Diskette! Da steht uns noch eine Menge Arbeit bevor!

Nach zähem Ringen gelingt es mir, eine Version für wahre Anfänger wie mich zu erstellen. Wie soll meine erste eigene Diskette heißen (Name)?

Probe aufs Exempel

Der Name kann bis zu 16 Zeichen lang sein. Ich nenne sie »HURRA«. Jetzt muß ich zwei Ziffern als Nummer meiner Diskette aussuchen. Das ist sehr wichtig, anhand dieser Nummer identifiziert das Laufwerk die Diskette. Der Name ist für sie unwichtig. Er dient nur meiner eigenen Orientierung. Die Diskette »YEP YEP, 01« ist für das Laufwerk dieselbe, wie die mit dem Namen »HALLO, 01«. Ich wähle die Identitätsnummer 01 (007 geht leider nicht). Hier die wahre Version für alle Anfänger wie mich!

```
OPEN 15,8,15,"N:HURRA,01"
```

Nach dem Eingeben der Zeile (natürlich mit [RETURN]) rattert und surrt das Disketten-Laufwerk los. Der Computer meldet sich mit »READY«, die Disketten-Station arbeitet noch, da die grüne Leuchtdiode leuchtet. Nach einer Weile geht sie aus und ich beratschlage mit meinem Denkapparat, was zu tun ist. Ich gebe CLOSE 15 ein und drücke [RETURN]. Die Veränderungen auf dem Bildschirm sind umwerfend, READY erscheint und der Cursor rückt eine Zeile weiter. Was soll das alles? Was haben die Zahlen und Buchstaben des Formatierbefehls für eine Bedeutung?

Von OPEN 15,unit,15,"N(drive):name,id" zu OPEN15,8,15,"N:HURRA,01". Am einfachsten ist die Ziffer 8, da sie schon bekannt ist. »Unit« heißt auf deutsch Einheit und bezieht sich auf ein an unseren Computer angeschlossenes Peripheriegerät. Peripheriegeräte werden alle in der Umgebung (Peripherie gleich Umgebung) angeschlossenen Geräte genannt. Mein einziges Peripheriegerät ist die Disketten-Station mit Nummer 8. Jedes angeschlossene Gerät erhält vom Computer eine Nummer, so weiß er immer, um welches Gerät es sich handelt. Die Floppy-Station hat die »Hausnummer« 8. Diese Zahl kennt Ihr schon vom SAVE-Befehl: SAVE "HENNING",8 gab der Station den Befehl, das Programm zu speichern. Die erste 15 hat für uns noch keine große Bedeutung. Dafür aber die zweite. Sie und der Befehl OPEN gehören zusammen. Die 15 ist ein Symbol für einen Kanal, auf dem die Disketten-Station ganz bestimmte Befehle empfängt. In der Disketten-Station befindet sich ein kleiner Computer. Er empfängt Befehle und Daten vom C 64. Die Daten müssen irgendwie zwischen beiden Geräten ausgetauscht werden. So wie PKWs auf Straßen hin- und herfahren, pendeln Daten zwischen Computer und Disketten-Station in Kanälen. 16 Kanäle stehen dem C 64 für den Datenverkehr mit der Disketten-Station zur Verfügung. Kanal 15 hat eine Sonderstellung. Über ihn werden ausschließlich Befehle gesendet, er ist also ein reiner Befehlskanal. OPEN 15,8,15 heißt also »Befehl an das Gerät mit der Nummer 8, Kanal 15 öffnen, um Informationen zu schicken«. Die erwarteten Informationen sind der Befehl <N> zur Formatierung, der Name HURRA und die Identitätsnummer der Diskette. CLOSE 15 ist der Befehl, den Kanal wieder zu schließen. Kanäle müssen immer geschlossen sein, wenn sie nicht gebraucht werden. Diese Vorgänge spielen sich zwischen dem Computer und dem Disketten-Laufwerk ab und sind auf dem Bildschirm nicht sichtbar!

Das war der dickste Brocken und nun zurück zu dem Problem, das den Rundumschlag erst nötig machte. Die Diskette ist formatiert, kann ich jetzt mein Programm laden?

Nachdem ich den Speicher des Computers mit NEW und [RETURN] gelöscht habe, gebe ich das Programm erneut ein. Das Speichern verläuft ohne Probleme, der Computer meldet sich mit SAVING HENNING und das Disketten-Laufwerk summt vor sich hin. Das Programm ist geladen. Tja, nun ist mein Programm auf Diskette gespeichert. Ein bißchen unsicher bin ich doch.

Als ängstlicher Mensch frage ich mich: Was ist, wenn der Computer falsch gearbeitet hat, oder meine Diskette einen kleinen Defekt aufweist? Morgen setze ich mich vielleicht frohgemut an meinen Computer und will mit dem abgespeicherten Programm arbeiten, doch es funktioniert nicht! Ich gebe zu, mein Programm ist nicht gerade umwerfend lang. Der Verlust wäre zu verkraften und ohne weiteres zu ersetzen, aber bei langen Programmen kann das übel enden. Aus diesem Grunde gibt es den Befehl VERIFY. Er vergleicht die im Speicher des Computers befindlichen Daten mit den soeben auf Diskette übertragenen. Bei einer Unregelmäßigkeit fängt der C 64 zu meckern an, wenn alles klar geht, sind die Daten richtig gespeichert. Die Sache funktioniert folgendermaßen: Im Moment ist das Programm noch im Arbeitsspeicher des Computers. Ich gebe ein:

```
VERIFY "HENNING",8    <RETURN>
```

Wieder legt das Disketten-Laufwerk los. Auf dem Bildschirm erscheint

```
SEARCHING FOR HENNING
VERIFYING
OK

READY
```

Der Computer hat sich mit OK gemeldet, das Programm ist korrekt abgespeichert worden. Durch den VERIFY-Befehl habe ich die Sicherheit, daß alles gut verlaufen ist. Hätte sich unser Computer-Kumpel mit

```
?VERIFY ERROR
```

gemeldet, so wäre ein Fehler unterlaufen. In diesem Falle hilft nur erneutes Speichern, mit VERIFY kann alles überprüft werden.

Formatieren - alles klar?

Trotzdem sind noch Fragen offen. Wieso soll ich einer Diskette einen Namen und eine Identifizierungsnummer geben, die ich hinterher nicht sehe? Die Sache ist doch sehr rätselhaft.

Es gibt einen kleinen Haken. Der Formatierbefehl beinhaltet mehr als nur ein Auftragen der Spuren und Sektoren. Auf der Suche nach meinen HURRA 01-Informationen bin ich auf folgenden Befehl gestoßen: LOAD"$",8. Das Zeichen zwischen den Anführungsstrichen steht für eine Art Inhaltsverzeichnis der Diskette. LOAD "$",8 und dann LIST sollen Namen und Inhaltsverzeichnis auf den Bildschirm bringen. Achtung! Dieser Befehl löscht die im Speicher des Computers enthaltenen Programme! Mein HENNING-Programm wird von LOAD"$",8 aus dem Speicher geworfen und verschwindet in den ewigen elektronischen Jagdgründen. Aber ich habe es ja noch auf Diskette!

Da steht es vor mir, »HURRA 01«, sogar in einem weißen Balken (Bild 3.2). Ich bin äußerst zufrieden mit mir und vergesse jeglichen alten Streit mit meinem Computer. Wir sind ein Herz und eine Seele. Das Inhaltsverzeichnis meiner Diskette besteht außerdem aus dem Programm HENNING und einigen Wörtern und Ziffern, deren Bedeutung mir noch schleierhaft ist.

Bild 3.2: LOAD"$",8 ruft das gesuchte Inhaltsverzeichnis auf den Bildschirm.

Blöcke vorm Kopf

Bei der Formatierung der Diskette entstehen 664 leere Blöcke, die zum Speichern zur Verfügung stehen. Die Zahl vor dem HENNING-Programm gibt die Anzahl der bei der Speicherung belegten Blöcke an. Die Zeile »664 BLOCKS FREE« listet die freien Blöcke einer Diskette auf (BLOCKS FREE: deutsch »freie Blöcke«). Dieses Informa-

tionssystem nennt sich BAM (Block Availability MAP – Blockbelegungsplan). Ich habe immer einen Überblick über den noch zur Verfügung stehenden Speicherplatz.

Was diese kleinen schwarzen Scheiben alles aufnehmen können: Spuren, Sektoren und jetzt das BAM. Bei aller Bewunderung: Kann eine Diskette die vom Computer kommenden Informationen selbst verarbeiten und die Zahl der freien Blöcke erkennen?

Die Antwort liegt beim Disketten-Laufwerk. Das Laufwerk ist ein kleiner »Computer« für sich. Durch bestimmte Signale tritt dieser Computer in Aktion und stellt zum Beispiel die freibleibenden Blöcke dar.

Die Diskette ist das Blatt, auf das die Daten geschrieben werden.

Dieses »einfache« Blatt ist genau eingeteilt. Eine Besonderheit ist die Spur 18, denn hier befindet sich das BAM. Die an die Diskette übermittelten Daten benötigen einen Platz, an dem sie gespeichert werden können und der nie mit anderen Daten belegt wird.

Bei LOAD"$",8 fährt das Disketten-Laufwerk zuerst diesen Abschnitt an.

Es liest die dort gespeicherten Daten ab und gibt sie als Inhaltsverzeichnis an den Bildschirm weiter. In dieser Spur erfährt das Disketten-Laufwerk auch, wo und in welchen Blöcken welche Programme gespeichert sind.

Ich lehne mich zurück und denke über die bisherigen Erkenntnisse nach. Schon eine Weile beschäftigt mich eine Frage. Wie sehen diese sagenumwobenen Spuren eigentlich aus, von denen ich ständig lese? Durch die Probleme mit der Formatierung ist diese Lücke in meinem Wissen bisher nicht aufgefallen.

Jetzt ist mein Interesse geweckt. Vorsichtig lege ich den Sicherungshebel um und hole die Diskette aus dem Laufwerk heraus, halte sie an der schwarzen Schutzhülle und betrachte die freiliegenden Partikel der beschichteten Folie (Bild 3.3). Richtige Rillen oder Spuren wie bei einer Schallplatte sind nicht zu erkennen.

Wie funktioniert es?

Bild 3.3: Wichtige Begriffe rund um die Diskette. Wenn die Schreibschutzkerbe überklebt wird, kann die Diskette nur noch »gelesen« werden.

Die Folie sieht recht unscheinbar, allerdings auch sehr empfindlich aus. Genau gehe ich noch einmal durch, wie ich mit Disketten umgehen soll, Vorsicht ist geboten.

- Niemals die Diskette an der Oberfläche der freiliegenden Folie berühren.
- Niemals die Diskette aus dem Schutzumschlag entfernen, wenn sie nicht sofort verwendet wird.
- Niemals die Diskette mit Kugelschreibern oder spitzen Bleistiften beschriften, der Aufpreßdruck des Stiftes ruiniert die gespeicherten Daten!
- Niemals die Diskette biegen, falten oder knicken.
- Niemals schwere Gegenstände auf die Diskette stellen oder legen.
- Niemals die Diskette mit Flüssigkeiten reinigen, auch nicht mit reinem Alkohol. Ihr riskiert die Auflösung der Datenträgerschicht!
- Niemals die Diskette großer Hitze aussetzen.
- Niemals die Diskette bei Temperaturen unter 10 Grad einsetzen oder lagern.
- Niemals die Diskette mit Gewalt in die Disketten-Station einführen.
- Niemals die Diskette auf elektrische Geräte wie Fernseher oder Radio legen.

- Niemals die Diskette als Frisbee-Scheibe verwenden.
- Die Diskette immer aus dem Laufwerk entfernen, wenn das Gerät abgeschaltet werden soll.
- Disketten immer aufrecht stehend lagern.
- Disketten immer an der Oberseite anfassen. Vorsichtig ins Laufwerk einführen!

Überall lauern Feinde

Die Empfindlichkeit der Diskette ist in der magnetischen Beschichtung der Kunststofffolie begründet. Sie besteht aus kleinen magnetisierbaren Partikeln. Beim Formatieren oder Speichern magnetisiert der Schreib-/Lesekopf die Beschichtung und »prägt« dadurch Daten auf die Magnetschicht. Das Ergebnis dieses Vorgangs ist auf der Diskette nicht sichtbar.

Besonders gefährlich sind magnetische Felder. Einen Magneten kennt jeder. Er besteht aus eisenanziehendem Stahl. Fast jedes elektrische Gerät, sei es der Fernseher oder der Kassettenrecorder, produziert beim Betrieb magnetische Felder. Sie können die auf einer Diskette gespeicherten Daten vernichten, da die Beschichtung »ummagnetisiert« wird. Bei längeren Programmen kann die Veränderung eines einzigen Blocks den Verlust der gesamten Diskette bedeuten. Also ist äußerste Vorsicht geboten. Das gleiche passiert natürlich bei Fett- oder Dreckflecken auf der Diskette, da der Lesekopf an seiner Arbeit gehindert wird. Um das Risiko möglichst klein zu halten, empfiehlt es sich, von wichtigen Disketten eine Sicherheitskopie zu machen.

Leseübung durch ein ovales Fenster

Eine letzte Frage wartet noch auf Antwort. Was passiert in der Disketten-Station? Woher rühren die merkwürdigen zackigen Geräusche? Die Antwort ist nicht schwer. Durch das Kopffenster (siehe Bild 3.3) hat der Lesekopf der Disketten-Station freien Zugang zur Diskette. Da sie mit mehreren Umdrehungen pro Sekunde gedreht wird, muß der Kopf nur die einzelnen Spuren abfahren, daher die ruckartigen Geräusche.

Oh, diese Theorie! Mir brummt längst der Schädel. 1571, Lesekopf, BAM, OPEN 15... Trotzdem: Bei meinem nächsten Gespräch kann ich mitfachsimpeln, hier etwas über die Einwirkungen von Magnetfeldern auf Disketten beitragen oder dort etwas über die BAM auf Spur 18 und Befehlskanal 15.

Ach ja, ein Thema will ich noch ansprechen: die verschiedenen Disketten-Arten. Im Kaufhaus konnte ich mich von der Fülle der Sorten überzeugen. Der Verkäufer überfiel mich mit Begriffen wie »DD«, »SD«, »SS« oder »DS«. Nun, dieses Problem konnte sehr schnell aus der Welt geschafft werden. »DD« steht für Double Density (zu

deutsch: doppelte Schreibdichte), »SD« für Single Density (einfache Schreibdichte). Für unsere 1571- oder 1541-Laufwerke reichen letztere vollkommen aus. Die beiden anderen Bezeichnungen beziehen sich auf die beschreibbaren Seiten der Diskette. »DS« bedeutet Double Sided (zweiseitig) und »SS« (einseitig). Wir verwenden in der Regel einseitige Disketten. Der Gebrauch von zweiseitig beschriebenen Disketten ist ganz einfach. Die Diskette muß nur umgedreht und wieder in das Laufwerk eingeschoben werden.

Das haben wir gelernt

Diskette: Eine Magnetscheibe, auf der Daten gespeichert werden können. Sie ist von einer Schutzhülle umgeben.

Formatieren: Eine neue Diskette muß betriebsbereit gemacht (formatiert) werden, sonst können wir nicht mit ihr arbeiten.

OPEN: Öffnet einen Befehlskanal. Mit dem Befehl OPEN,8,15,8,"N:Name,id" wird eine Diskette formatiert. Für »Name« wird der gewünschte Name eingegeben, »id« ist die Identifizierungsnummer. Zum Beispiel:

```
HURRA,01
```

VERIFY: Überprüft die gespeicherten Daten mit denen im C 64. Dieser Befehl arbeitet auch mit der Datasette.

Kapitel 4

Der C 64 als Rechengenie

Das Rechnen ist des Müllers Lust. Unser Computer ist ein wahres Rechengenie. Er kann einfache Rechenoperationen mit den vier Grundrechenarten genauso ausführen wie komplizierte mathematische Funktionen (zum Beispiel Trigonometrie oder Algebra). Hey Leute, laßt Euch von den Ausdrücken nicht schocken! Die Bedienung weicht etwas von der eines Taschenrechners ab, aber was der kann, kann der C 64 schon lange. Genau genommen ist der C 64 ein elektronischer Rechner, nur mit wesentlich mehr Möglichkeiten als die, die wir von der Schule her kennen. Beginnen wir mit den Grundrechenarten. Vorher müssen wir uns eines klarmachen: Wir benötigen ständig den Befehl PRINT. Er läßt den Computer die errechneten Zahlen unverzüglich auf den Bildschirm drucken. Ohne PRINT behält der C 64 die Werte für sich und davon haben wir überhaupt nichts. Nehmen wir uns die erste Rechnung vor.

Ich tippe ein:

```
5 + 5 =  <RETURN>
```

Tja, da bin ich doch reingefallen. Der liebe Computer tut genau das, was ich ihm gesagt habe: er rechnet 5 + 5, sieht aber überhaupt keine Veranlassung, mir den erhaltenen Wert mitzuteilen. Ich muß PRINT vor die ganze Geschichte setzen. Also nochmal das Ganze

```
PRINT 5 + 5  <RETURN>
```

Es erscheint:

```
PRINT 5 + 5  <RETURN>
 10

READY
```

Die Rechenzeile bedeutet: Rechne 5 + 5 und schreibe das Ergebnis dieser Rechnung auf den Bildschirm. Wir sehen, die Addition (das »Zusammenzählen«) beherrscht der

C 64 spielend. Wie steht es mit der Subtraktion (dem »Abziehen«)? Das Verfahren ist das gleiche.

```
PRINT 500 - 150  <RETURN>
 350

READY
```

Hausaufgaben ade!

Prima, prima. Nie wieder Mathe-Hausaufgaben! Die macht von jetzt an dieser faszinierende Kasten vor mir. Mit der Multiplikation (»Malnehmen«) und der Division (»Teilen«) funktioniert das Rechnen nach dem gleichen Schema. Eine Auflistung der verschiedenen Rechensymbole und Befehle mit ihrer Funktion findet Ihr in Tabelle 4.1 am Ende des Kapitels.

Mit diesen Zeichen und Befehlen in Zusammenhang mit PRINT können wir nach Lust und Laune herumspielen. Nur nach jeder Befehlszeile die RETURN-Taste drücken, denn erst durch [RETURN] wird dem Computer die neue Zeile mitgeteilt. In der Tabelle wird Euch aufgefallen sein, daß es neben den mathematischen Zeichen auch Befehle gibt. Der C 64 ist in der Lage, mathematische Operationen auszuführen, für die er keine Zeichen besitzt. Das Zeichen zum Wurzelziehen ist zum Beispiel im Zeichensatz des C 64 nicht vorhanden. Dafür gibt es den Befehl SQR(X). X ist ein Platzhalter, es muß mit einer Zahl ersetzt werden.

```
PRINT SQR(9) <RETURN>
```

hat das richtige Ergebnis 3. Probiert es aus und macht Euch ruhig mit den anderen Funktionen vertraut. Einigen sind viele Funktionen vielleicht nicht ganz klar. Das macht nichts. Sie sind zum Beispiel für wissenschaftliche Rechnungen von Bedeutung. In der Schule werden sie in höheren Klassen besprochen. Dort finden diese Rechenoperationen in Kurvendiskussionen Anwendung. Wer in der Oberstufe ist und dieses Thema gerade durchnimmt, kann sich mit ein wenig mehr Programmier-Erfahrung ein Programm schreiben, das eine komplette Kurvendiskussion ausführt. Doch im Moment ist das für uns etwas zu hoch gegriffen. Wenden wir uns deshalb den Grundlagen zu.

Ein kleiner Tip: Wenn Euch das ewige Eintippen von PRINT auf den Wecker geht, probiert anstelle von PRINT doch mal ein Fragezeichen aus. Das Fragezeichen ist die Abkürzung für PRINT und kann eine Menge Arbeit ersparen, auch später beim Programmieren:

```
? 100 + 5 - 68 + 20/10
```

ist das gleiche wie

```
PRINT 100 + 5 - 68 + 20/10.
```

Für unser weiteres Fortkommen am C 64 müssen wir eines beachten: Der C 64 ist und bleibt ein amerikanisches Gerät! Die Amerikaner schreiben nicht 600,88, sondern 600.88! Da, wo wir normalerweise ein Komma setzen (die sogenannten Kommazahlen), verwenden die Amerikaner den Punkt. Probiert mal folgendes aus:

```
? 10,5 + 2,5  <RETURN>
```

Es erscheint:

```
10        7        5
```

Da sind wir ganz schön angeschmiert, was? Das Komma hat in Basic-Befehlszeilen die Funktion eines »Tabulators« (wir begegnen ihm noch genauer im Rahmen unseres Basic-Kurses). Unsere Befehlszeile zeigt dem Computer eine auszudruckene Zeile an. In dieser Zeile befinden sich für den Computer drei verschiedene Werte, die er drucken soll. Sie sind durch Kommas getrennt, also behandelt er sie auch so. Zunächst druckt er die 10 aus, dann rechnet er bis zum nächsten Komma 5 + 2 = 7 und schreibt den Wert ein Stück rechts von der 10. Als letztes wird einfach die dritte Zahl geschrieben, die 5. Die Kommas bewirken, daß die drei Zahlen in eine Zeile, allerdings mit Abstand (tabelliert) geschrieben werden. Wir dürfen Komma und Punkt nie verwechseln, sonst gehen die schönsten Rechnungen in die Hose. Gebt mal die folgende Zeile ein:

```
PRINT 10.5 + 2.5
```

Das Ergebnis ist 13 und damit richtig.

Für eine Handvoll Rechnen mehr

Es gibt auch kompliziertere Rechnungen! Bei der Vermengung von Punkt- (Multiplikation und Division) und Strichrechnungen (Addition und Subtraktion) müssen wir ebenfalls aufpassen. Die Zeile

```
PRINT 12 + 4 * 5 - 8
```

ergibt für den Rechner 24 und nichts anderes. Der Computer führt zuerst alle Punktrechnungen aus. Das heißt, er rechnet 4*5=20 und setzt das Ergebnis an die richtige Stelle. Nun erst führt er die anderen Rechnungen aus: 12+20−8=24.

Kommen wir nun zu einer mathematischen Spezialität: den Potenzzahlen. Unter einer Potenzzahl versteht man eine Zahl, die mehrfach mit sich selber malgenommen wird. Zwei hoch zwei bedeutet: 2*2. Zehn hoch fünf heißt: 10*10*10*10*10. Die Basiszahl, das ist die Zahl, die »unten« steht, wird so oft mit sich selber malgenommen, wie es die Hochzahl angibt.

Wir dürfen uns aber nicht wundern, wenn der C 64 bei sehr großen Zahlen regelrechte Ungetüme produziert. Die größte Zahl, die er »normal« ausgibt, ist 999999999. Um Euch das Nachzählen zu ersparen, es handelt sich um eine Zahl mit neun Neunen. Die nächsthöhere Zahl druckt er ganz anders aus. Schreibt

```
PRINT 999999999+1
```

Der C 64 druckt

```
1E+09
```

Das große E ist die Abkürzung für »10 hoch«. Die Zeile muß so gelesen werden: »Einmal 10 hoch 9«, also:

```
10*10*10*10*10*10*10*10*10=1000000000.
```

Die Zahl 1.4567E+08 wäre also für uns: »1,4567 multipliziert mit 10 hoch 8.« Es ist schon ein bißchen gewöhnungsbedürftig, oder nicht? Überall da, wo das große E steht, müßt Ihr »10 hoch« einsetzen.

Bei zu großen Zahlen fängt der C 64 an zu mosern. Die Zeile

```
?100E+50
```

quittiert er mit

```
?OVERFLOW ERROR
READY
```

Sie ist ihm zu groß. OVERFLOW ERROR heißt auf deutsch soviel wie »Zahlenüberlauf«. (Hier eine wichtige Information für später: Die Darstellung von »normalen« Hochzahlen erfolgt mit Hilfe der Pfeil-nach-oben-Taste. Auf ihr befindet sich ein nach oben gerichteter Pfeil).

Natürlich kann der C 64 auch Bruchrechnungen ausführen. Wollen wir zum Beispiel die Wurzel von 10 durch die Wurzel von 5 minus 1 teilen, muß es wie folgt geschrieben werden:

```
PRINT SQR(10)/(SQR(5)-1)
```

Das bedeutet, Nenner (so wird der Teil oberhalb des Bruchstriches genannt, beim C 64 ist es der Teil vor dem Schrägstrich) und Zähler (der untere Teil, nach dem Schrägstrich) eines Bruches müssen in Klammern zusammengefaßt werden, wenn sie mehr als eine Zahl enthalten. In unserem Beispiel ist das Ergebnis 2,55833637. Schreiben wir die Rechnung so:

```
PRINT SQR(10)/SQR(5)-1
```

lautete das Ergebnis 0,414213562. In diesem Falle rechnet der C 64 nämlich erst Wurzel 10 geteilt durch Wurzel 5. Von dieser Zahl zieht er dann 1 ab. Deshalb Vorsicht!

Wenn man nicht aufpaßt, kann man in Teufels Küche geraten. Diese Fehler sind schwer zu finden und können ein Programm ganz schön durcheinanderbringen.

Bisher haben wir nur gerechnet, ohne unsere Rechnungen sehen zu können. Das ändert sich mit einem neuen Trick. Wir verbinden die PRINT-Anweisung mit Worten und Rechnungen! Zum Beispiel:

```
PRINT "2 + 2=",2 + 2  <RETURN>
2 + 2= 4
```

Ha, da haben wir eine Rechenaufgabe so, wie wir sie kennen. Der Rechner schreibt nicht nur das Ergebnis, sondern auch die Aufgabenstellung selber hin. Wie macht er das? Durch die Doppelbenutzung des PRINT-Befehls. PRINT schreibt die in Anführungszeichen gesetzten Zeichen vollständig und ohne Veränderung auf den Bildschirm. Die Befehlszeile beinhaltet zwei Befehle. Es soll auch der Wert von 2+2 ausgerechnet werden (hinter dem Semikolon). Das PRINT gilt auch für diesen Teil des Befehls. Der Computer schreibt zuerst den in Anführungszeichen stehenden Ausdruck, rechnet dann 2+2 und schreibt dieses Ergebnis ebenfalls auf den Bildschirm. Prima was? Wir sind ganz schön geschickt. Wir haben jetzt das Rüstzeug für so manch schwierige Aufgaben.

+	Addition
-	Subtraktion
*	Multiplikation
/	Division
↑	Potenzieren
ABS(X)	Der absolute Wert einer Zahl X
ATN(X)	Der Winkel im Bogenmaß mit dem Tangens X
COS(X)	Der Cosinus des Winkels X (Bogenmaß angeben)
EXP(X)	Die X-te Potenz der mathematischen Konstanten e (= 2,71827183)
LOG(X)	Logarithmus von X zur Basis e
SGN(X)	Ermittelt Vorzeichen der Zahl X
SIN(X)	Ergibt den Sinus des Winkels X (Bogenmaß)
SQR(X)	Quadratwurzel von X
TAN(X)	Tangens des Winkels X (Bogenmaß)

Tabelle 4.1: Der C 64 besitzt neben mathematischen Zeichen Befehle mit komplizierteren Funktionen

Das haben wir gelernt

Rechner: Der C 64 ist im Prinzip ein gigantischer Taschenrechner. Mit ihm können wir viele verschiedene Rechnungen durchführen.

?: Die Abkürzung für den Befehl PRINT.

E: Die Kurzdarstellung für eine Potenz zur Basis 10. 1E+09 wird 10 hoch 9 gelesen, 1E-09 hieße 10 hoch minus 9.

Kapitel 5

Schritte in Schleifen

Jetzt geht es rund! Basic in Sichtweite! Dabei komme ich an Schuhkartons voller Dollarzeichen und merkwürdigen Zeichengebilden vorbei. Ich sehe Zeiten nahen, in denen ich mitreden kann, wenn es um Schleifen, IF...THEN oder FOR...NEXT geht! Gleich wagen wir uns in die Tiefen der Programmierspache Basic!

Wie war das doch gleich?

Mein erstes Computer-Programm hieß HENNING. Es ließ den Computer so oft ich wollte die Worte »HENNING LEGT LOS« schreiben. Ich überprüfe meine Kenntnisse, indem ich mir ein neues Programm (in der Fachsprache »Listing«) erarbeite. Für »HENNING PACKT AUS« setze ich »AUF ZU NEUEN ABENTEUERN« ein. Es lautet:

```
10 PRINT "AUF ZU NEUEN ABENTEUERN"
<RETURN>
20 GOTO 10
<RETURN>
```

Ich starte es mit RUN [RETURN] und der Computer schreibt befehlsgemäß immer wieder »AUF ZU NEUEN ABENTEUERN«. Nach einer Weile habe ich mein Werk genug betrachtet. Ich unterbreche das Programm mit der RUN/STOP-Taste. Da muß doch mehr dahinterstecken als nur dieser Zweizeiler.

Es gibt einen Programmier- und einen Direkt-Modus. Im ersten Kapitel haben wir den PRINT-Befehl kennengelernt. Im nachhinein fällt mir auf: Manchmal stand vor dem Befehl eine Zahl, manchmal nicht. Wie kommt das? Die Zeile

```
PRINT "AB GEHT DIE POST"
```

wird nach dem Drücken der RETURN-Taste sofort ausgeführt, während der Befehl

```
10 PRINT "AB GEHT DIE POST"
```

erst nach der Eingabe von RUN in Aktion tritt. Der Schlüssel zum Geheimnis ist der Direkt-Modus. Wie der Name schon sagt, werden die im Direkt-Modus eingegebenen Befehle direkt nach dem Drücken der RETURN-Taste ausgeführt. Im Programmier-Modus wird eine Befehlszeile erst einmal in den Arbeitsspeicher des C 64 gelegt und auf weitere Befehlszeilen gewartet. Der Computer erwartet im Programmier-Modus ein Basic-Programm von mehreren Zeilen und nicht nur eine einzige Zeile wie im Direkt-Modus. Der C 64 schaltet automatisch in den Programmier-Modus um, sobald vor einer Befehlszeile eine Zahl steht! Das ist des Rätsels Lösung. Im zweiten Beispiel steht die Zeilennumerierung 10 vor der PRINT-Zeile. Der C 64 erkennt ein Basic-Programm, wartet auf weitere Eingaben und schaltet automatisch in den Programmier-Modus um.

Bäumchen, Bäumchen wechsel dich

Fehlerhafte Befehlszeilen können ohne Probleme ausgetauscht werden. Der C 64 ist ein schlaues Kerlchen. In meinem Programm ist jede Zeile mit einer Nummer versehen, so erkennt der C 64 immer die richtige Reihenfolge der Befehle. Angenommen, die erste Zeile (Nummer 10) meines Programms wäre fehlerhaft oder gefiele mir nicht mehr. Ich tippe ein:

```
10 PRINT "HENNING LERNT BASIC"   <RETURN>
```

und starte mein Programm mit RUN und [RETURN]. Der Stolz des Fachmanns erfüllt mich, denn der C 64 schreibt immer wieder »HENNING LERNT BASIC«, er hat die Zeilen einfach ausgetauscht. Wie funktioniert das?

Ein eingetipptes Programm wird im Arbeitsspeicher des Computers festgehalten. Meine neue Befehlszeile hat die Nummer 10. Diese beiden machen sich den Platz streitig. Die HENNING LERNT BASIC-Zeile hat Vorrang, da sie später eingegeben wurde. Der Computer wirft die alte Zeile aus dem Speicher, mein neues Programm ist einsatzbereit. Vorsicht: Der Speicher wird gelöscht, wenn

1. NEW und [RETURN] eingetippt wird,
2. der Computer ausgeschaltet wird und
3. das Inhaltsverzeichnis einer Diskette abgerufen wird.

Dann sind alle frisch eingetippten Programme futsch!

Mit diesem Verfahren kann man Zeilen löschen oder einige Zeichen in dem jeweiligen Befehl verändern. »10« und [RETURN] zum Beispiel löscht die Zeile 10.

Ich kann auch mit dem Cursor eine Befehlszeile anfahren und verändern, hinterher immer [RETURN] drücken! Das ist besonders wichtig, denn erst durch [RETURN] erkennt der C 64 die neue Befehlszeile. Alles kein Problem. Mit ein wenig Übung schreibt mein C 64, was ich will. Weiter im Text.

Oh manoman, da kommt der Hammer auf uns zu. Ich sehe Variablen mit Dollar- und Prozentzeichen in Schuhkartons vor uns! Etwas kompliziert! So schnell lassen wir uns nicht ins Bockshorn jagen, jetzt mal die Sache ganz von vorne aufrollen.

Fächerturm

Variablen sind für unsere Arbeit mit dem Computer besonders wichtig. Der Begriff Variable ist einigen vielleicht aus dem Mathematikunterricht bekannt. Das sind die X und Ypsilons, die in den Formeln auftauchen (zum Beispiel: 2=X+1). In der Mathematik sind es Platzhalter für Zahlen. Beim Computer haben sie einige Funktionen mehr. In einem Programm kommt es oft vor, daß der Computer sich irgendwelche Zahlen oder Texte »merken« muß.

»Merken« bedeutet, er legt sie irgendwo im Speicher ab und kann sie jederzeit dort wieder herausholen, vorausgesetzt, der Programmierer kennt ihren Namen. Variablen sind Namen, unter denen sich der C 64 Zahlen oder Texte merkt.

Ich stelle mir den Speicher meines Computers als einen Raum mit drei großen Regalen vor (siehe Bild 5.1). In jedes Regal kann ich Zettel legen. Es gibt drei verschiedene Sorten Zettel:

1. Zettel mit ganzen Zahlen.
2. Zettel mit Kommazahlen.
3. Zettel mit Zeichenketten.

Zeichenketten (oder auch String-Variablen, englisch String: »Schnur«) bestehen aus einer Aneinanderreihung von Buchstaben, Zahlen oder sonstigen Zeichen.

Ganze Zahlen (Integer-Zahlen) sind alle Zahlen ohne Komma. Kommazahlen sind alle Zahlen mit Komma (auch Fließkommazahlen genannt).

Bild 5.1: Wie in Schubladen von Regalen legt der C 64 die Variablen-Inhalte in seinem Speicher ab.

Jedes Regal hat mehrere Schubladen. Die Schubladen enthalten die verschiedenen Variablen-Inhalte. In der obersten ist zum Beispiel der Inhalt der Variablen A$ enthalten und in der unteren der der Variablen ZR$ (Das $-Zeichen findet Ihr auf eurer Tastatur über der 4). In Tabelle 5.1 habe ich mir einige Beispiele für mögliche Inhalte meiner Schubladen aufgelistet.

```
Integer-Zahlen          5   267      5800        6521     20978
Fließkommazahlen      3.7   25.86     896.34   27456.78
String-Variablen     COMMODORE + X%        KARL-HEINZ
                   DIE SUMME VON X + X% =
     (Achtung: Das sind drei verschiedene Zeichenketten!)
```

Tabelle 5.1: Einige mögliche Variablen-Inhalte.

Zettelwirtschaft

Bisher habe ich mich um die Inhalte der Schubfächer gekümmert, was ist mit den Schubfächern selbst?

Der Computer muß von außen erkennen, welche Art von Inhalt ein bestimmter Speicherplatz (Schublade) enthält. Deshalb gibt es drei verschiedene »Aufkleber«.

Die kleine Tabelle 5.2 hilft hier sehr, auch um später einmal nachzusehen.

Inhalt	Bezeichnung
Integer-Zahl	% am Ende des Namens
Fließkommazahl	Kein besonderes Zeichen
Zeichenkette	$ am Ende des Namens

Tabelle 5.2: Die Bezeichnungen der Variablen-Typen.

Wie ist diese Tabelle nun zu verstehen? Angenommen, ich will einer Integer-Zahl einen bestimmten Speicherplatz (Schublade) zuweisen. Ich stecke sie in irgendeine Schublade des Regals für Integer-Zahlen. Was mir fehlt, ist der Aufkleber. Ich denke mir einen Namen aus und schreibe ihn auf den Aufkleber: AB%. Wenn ich an meinen Computer-Regalen vorbeigehe, weiß ich: Dieser Speicherplatz heißt AB und enthält eine Integer-Zahl!

Beispiele für beliebige Namen von Variablen findet Ihr in Tabelle 5.3.

Integer-Variable	AB%	C1%	LD%	S8%	K%	Z%
Fließkomma-Variable	DE	F	K	SR	LM	H1
String-Variable	B$	D$	I9$	KI$	MT$	R5$

Tabelle 5.3: Beispiele für Variablen-Namen.

Zwei Dinge müssen beachtet werden:

1. Das erste Zeichen muß ein Buchstabe sein und der Name sollte nicht mehr als zwei Zeichen lang sein (% oder $ zählen nicht!). Die Variable HALLO$ ist für den C 64 dasselbe wie HA$.
2. Es dürfen keine Wörter verwendet werden, die in Basic eine bestimmte Funktion haben. Das sagt uns vorerst wenig, deshalb mein Tip: Falls der Computer irgendwelchen Ärger macht, nehmt einfach einen Namen aus Tabelle 3.

Alles Käse

Was soll das Ganze? Wieso raucht mir der Schädel von irgendwelchen Variablen? Wozu soll ich Zahlen in Schubladen stecken?

Denken, denken, denken. Am verständlichsten ist wieder einmal ein Beispiel. Leute, wir nehmen uns jetzt ein kleines Basic-Programm vor. Zuerst einmal abtippen, mit allen Semikolons und Kommas.

```
NEW
10 A% = 13
20 A  = 10.5
30 A$ = "DIE SUMME VON A% + A = "
40 PRINT "A% = "; A%, "A = ", A
50 PRINT A$; A% + A
```

Dann gebe ich RUN [RETURN] ein. Phantastisch, alles geht glatt, es erscheint:

```
A% = 13    A = 10.5
DIE SUMME VON A% + A = 23.5
```

Ich muß mir jede Zeile dieses Programms genau ansehen. NEW ist ein Basic-Befehl. Nach seiner Eingabe werden alle im Basic-Speicher enthaltenen Daten gelöscht. Deshalb aufpassen mit diesem Befehl. Eine vorschnelle Eingabe macht stundenlange Arbeit zunichte.

Die ersten drei Zeilen haben einen sehr ähnlichen Aufbau. So langsam dämmert es mir! Ich wußte vorhin schon nicht, wie ich meinem elektronischen Kumpel die Sache mit den Variablen klarmache. Ich kann ihm schlecht etwas von Zetteln und Schubfächern erzählen! Variablen und die dazugehörigen Werte werden mit Hilfe eines Gleichheitszeichens eingegeben. Nichts anderes geschieht in den Zeilen 10, 20 und 30. Ich ordne den verschiedenen Variablen-Typen Inhalte zu.

Scheibe für Scheibe

Zeile 10: Lege die ganze Zahl 13 in einen Speicher und nenne ihn A%. Für unsere Schubladen lautete der Befehl: Lege die Zahl 13 in eine Schublade des Regals für Integer-Zahlen und beschrifte sie mit »A%«.

Zeile 20: Lege die Fließkomma-Zahl 10.5 in einen anderen Speicher und nenne ihn »A«. (Vorsicht: Der Punkt nach der 10 ist weder Fliegendreck noch ein Druckfehler. Der C 64 ist eine Erfindung der Amerikaner, und die schreiben nicht 23,5 sondern 23.5. Wir müssen also bei Kommazahlen einen Punkt machen!)

Zeile 30: Lege die Zeichenkette »DIE SUMME VON A% + A = « in einen Speicher und nenne ihn »A$«.

Hier zeigt sich nun, wie wichtig der Umgang mit »%« und »$« ist. Alle drei Variablen beginnen mit A. Durch die beiden Anhängsel erkennt der Computer, daß es grundverschiedene Variablen sind und behandelt sie dementsprechend.

Rattenschwanz

Was bedeuten die merkwürdigen Zeichen in Zeile 40? PRINT bedeutet »schreibe«, das kennen wir. Die Anführungszeichen schließen den zu schreibenden Text ein. Der Computer schreibt »A% = «. Das folgende Semikolon heißt soviel wie: »Mache ohne Zwischenraum weiter«. Danach kommt A%. Achtung: Der C 64 schreibt nicht A%, hier sind keine Anführungszeichen vorhanden! A% ist die Bezeichnung eines Speicherplatzes! Er öffnet diese Schublade und schreibt deren Inhalt auf den Bildschirm, die Zahl 13. Nach dem gleichen Verfahren wird der zweite Teil der Zeile 40 behandelt, jetzt mit der Variablen A.

Zeile 40 bedeutet im Klartext: Schreibe »A% = « und direkt dahinter den Wert der Variablen A%; schreibe anschließend »A= « und direkt dahinter den Wert von A. Auf dem Bildschirm erscheint:

```
A% = 13    A = 10.5
```

Zeile 50 ist leicht zu verstehen: Drucke den Wert der String-Variablen A$ aus, addiere die beiden Variablen A% und A (13+10.5) und schreibe das Ergebnis hinter A$.

Der Computer druckt:

```
DIE SUMME VON A% + A=23.5
```

Leute, wir haben es geschafft! Wir haben uns durch dieses Programm gebissen und den Sinn verstanden. Das Programm ist nichts anderes als eine in die Computer-Sprache übersetzte Rechenaufgabe! Wir sind bereits mittendrin im Programmieren und lernen die »Gedankengänge« unseres C 64 kennen. So kompliziert ist das gar nicht, man muß sich nur Schritt für Schritt in ein Programm eindenken, und alles erscheint in logischen Zusammenhängen.

Neues ohne Ende

Im folgenden werden wir sehen, daß wir aus solchen einfachen Rechnungen mit einem kurzen Programm das ganze Alphabet ausdrucken können. Der Trick dabei ist: Zahlen werden zu Buchstaben und umgekehrt! Der Schlüssel zu dieser Verwandlung ist der Befehl CHR$ (sprich Character-String) und ASC$ (sprich ASCII-String). Diese beiden

Befehle eröffnen uns für spätere Programmier-Zeiten ein weites Feld an Verwendungsmöglichkeiten.

Jedes mit der Tastatur aufrufbare Zeichen kann in Form einer Zahl dargestellt werden! Es ist wie mit einer Geheimschrift, die Buchstaben und Zeichen in einen Zahlencode umwandelt. Die Zahl 65 zum Beispiel ist die Ziffer für den Buchstaben »A«. Ein solcher Zahlencode wird mit Hilfe des CHR$ wieder in ein Zeichen zurückverwandelt. CHR$ ist sozusagen die Decodierungsmaschine. Probieren wir aus:

```
PRINT CHR$ (65)    <RETURN>
```

Der Computer erfreut uns mit einem großen »A« in der nächsten Zeile des Bildschirms. Probiert mal ein wenig herum, »B« hat den Wert 66, »C« 67 und so weiter. Bei diesen Zahlen handelt es sich um sogenannte ASCII-Codes. ASCII bedeutet: »American Standard for Information Interchange« und heißt auf deutsch etwa: »Amerikanischer Standard für den Austausch von Informationen«. Eine Liste der gesamten ASCII-Werte findet Ihr hinten im Bedienungshandbuch. Die Bedeutung und der wahre Nutzen dieser Zahlen ist uns im Moment noch schleierhaft. Stellt Euch aber einmal die Verbindung von zwei Computern vor. Durch die ASCII-Codes ist eine genormte »Sprache« entwickelt worden, mit der sich Computer verständigen können. Beide Computer haben für den Buchstaben A denselben Zifferwert. Stellt Euch mal vor, ein Drucker hätte andere Codes für Zeichen als der an ihn angeschlossene Computer. Wir wären nicht in der Lage, einen vernünftigen Ausdruck zu machen.

Einen praktischen Nutzen für unsere jetzige Arbeit können wir uns sofort klarmachen. Probiert mal folgendes aus. Ihr schreibt zuerst PRINT und setzt dann ein Anführungszeichen. Nach dem Anführungszeichen drückt Ihr gleichzeitig die Tasten [SHIFT] [CLR/HOME]. Auf dem Bildschirm erscheint ein reverses Herz, jetzt setzt Ihr wieder ein Anführungszeichen und drückt [RETURN]. Der Bildschirm wird gelöscht, ganz oben erscheint das Wort READY. Mit der Kombination [SHIFT] [CLR/HOME] könnt Ihr jederzeit den Bildschirm löschen. In unserem Falle haben wir diese Tastenfunktion in eine Befehlszeile mit eingebunden. Das Zeichen für Bildschirmlöschen ist das reverse Herz.

```
PRINT "<SHIFT + CLR/HOME>"
```

Ein solches Zeichen in einem Programm ist natürlich etwas verwirrend. Stellt Euch vor, Ihr stoßt in einem Programm plötzlich auf ein reverses Herz und wißt nicht, wie Ihr es zustande kriegt! Theoretisch müßtet Ihr die ganze ASCII-Tabelle auswendig kennen, um gegen alle Probleme gewappnet zu sein. Nun stellt Euch aber einmal vor, in dem Programm steht statt obiger Zeile die folgende:

```
PRINT CHR$ (147)
```

Diese Zeile hat genau den gleichen Effekt wie die vorherige. Der Wert 147 ist der für das Bildschirmlöschen. Er ist nichts anderes, als das in ASCII-Code verpackte reverse Herz. Wenn Ihr in einem Programm auf eine solche Zeile stoßt, habt Ihr mit dem Ein-

tippen keine Probleme! Fassen wir zusammen: Alle Zeichen, nicht nur Buchstaben, lassen sich mit Hilfe von ASCII-Werten darstellen. In der Praxis erleichtern sie das Eingeben von Programmen und ermöglichen die Kommunikation von zwei Computern.

Die Geschichte funktioniert natürlich auch umgekehrt. Gebt folgende Zeile ein:

```
PRINT ASC("A")    <RETURN>
```

Auf dem Bildschirm erscheint der zum Buchstaben A gehörige ASCII-Wert 65. Oben haben wir aus einem Zahlenwert ein Zeichen gemacht, jetzt erhalten wir aus einem Zeichen einen Wert. Probieren wir unser neues Wissen an einem weiteren Beispiel aus. Wir schreiben

```
PRINT ASC("<SHIFT + CLR/HOME>")    <RETURN>
```

In der Klammer steht <SHIFT + CLR/HOME>. Hier drückt Ihr natürlich auf die angegebenen Tasten und gebt nicht die obigen Worte ein. Leider läßt sich das Herz nicht so gut darstellen, deshalb gebe ich die Tasten an, klaro?

Die Zahl 147 erscheint. Wie wir oben schon gesehen haben, ist dies der Wert für das Bildschirmlöschen. Soweit alles klar? Na, dann können uns Ausdrücke wie ASCII-Code von jetzt ab nicht mehr schocken.

Falsche Mathematik

Bevor wir zu unserem versprochenen Alphabet aus Zahlen kommen, müssen wir uns eine Gleichung ansehen, bei der mein Mathe-Lehrer seine letzten Haare verlieren würde. Ich habe auch einen großen Schrecken bekommen, als ich folgendes sah:

```
X=X+1
```

Was soll das nun wieder? Wie kann eine Zahl genauso groß sein wie die nächsthöhere? 4=4+1? An so etwas müssen wir uns gewöhnen, denn der Computer ist ein kleiner Schummler. Wenn er Lust hat, liest er von rechts nach links. Damit noch nicht genug, auch Variablen sind veränderbar! Gehen wir der Reihe nach vor.

Der einer Variablen zugeordnete Wert kann sich im Laufe eines Programms verändern. Durch X=X+1 zum Beispiel erhöht der Computer den Wert der in der Schublade X befindlichen Zahl jeweils um 1. Den neuen Wert legt er wieder in Speicher X ab, das Spiel beginnt von vorne.

Ich stelle mir vor, daß der Computer die Zeile »X=X+1« rückwärts liest. Den Anfangswert für X setzte ich gleich 1. Zuerst trifft der Computer auf X+1. Er rechnet: 1+1 und erhält 2. Auf der rechten Seite unserer Gleichung steht 2, auf der linken X. Für den Computer ist jetzt X=2, der Inhalt der Variablen hat sich verändert. Ein Beispiel für diese Arbeitsweise findet sich im folgenden Programm. Die einzige Neuheit ist der IF...THEN-Befehl, dessen Funktion uns schnell klar wird. Ich tippe ein:

```
NEW
10 X=65
20 PRINT CHR$(X);
30 X=X+1
40 IF X <> 91 THEN 20
50 END
```

Jetzt spielen wir ein bißchen Computer, dann ist das Programm ganz einfach.

Zeile 10: Ordne der Variablen X den Wert 65 zu.

Zeile 20: Gebe auf dem Bildschirm das Zeichen für den ASCII-Wert X aus. Im ersten Durchgang hat X den Wert 65. Der C 64 druckt A aus.

Zeile 30: Addiere zu X=0 den Wert 1 und ordne dem Ergebnis wieder die Variable X zu.

Zeile 40: Solange der Wert für X kleiner oder größer (nicht gleich) 91 ist, springe zu Zeile 20. Der Sprung nach Zeile 20 ist eine sogenannte »Schleife« (siehe Bild 5.2).

Bild 5.2: Der Sprung von Zeile 40 zu Zeile 20 ist eine Schleife.

Der Computer führt die Zeilen 20 bis 40 so lange aus, bis X gleich 91 ist. Bei jeder Schleife schreibt er einmal das Zeichen für den ASCII-Wert X, der sich mit jedem Mal um eins erhöht. Sobald X den Wert 91 erreicht, springe zur nächsten Zeile.

Zeile 50: Beende das Programm.

»IF...THEN« heißt auf deutsch, »wenn...dann«. Wir können innerhalb eines Programms Bedingungen stellen, die auf den Programm-Ablauf Einfluß haben. In unserem Fall schreibt der C 64 so lange ein neues Zeichen, bis X den Wert 91 angenommen hat. Wenn X den Wert 91 annimmt, ist die Bedingung, daß X ungleich 91 ist, nicht mehr erfüllt und die nächste Zeile wird ausgeführt.

Na, das war alles ganz schön neu, aber sonst nicht so schwer, wie es anfangs aussah. Jetzt können wir dem C 64 schon Bedingungen stellen. Ob es da nicht noch mehr Möglichkeiten gibt, ihm zu zeigen, wer der Herr im Hause ist? Folgendes Programm beantwortet unsere Frage mit »Ja«.

```
NEW
10 FOR A=1 TO 5
20 PRINT "ALLES PALETTI FREUNDE?"
30 NEXT A
```

Zeile 10: Zähle (im Verlauf des Programms) für A von 1 bis 5. Beginne mit A=1 und gehe dann in die nächste Zeile.

Zeile 20: Schreibe »ALLES PALETTI FREUNDE«

Zeile 30: Gehe zurück zu FOR (Zeile 10) und zähle das nächste A (zuerst also A=2). NEXT heißt auf deutsch: nächste, folgende.

Computerspiel

Klar, hier liegt wieder eine Schleife vor, diesmal zwischen FOR und NEXT. Die volle Befehlsfolge lautet FOR...TO...NEXT. Übersetzen wir die Befehle ins Deutsche, heißen sie FÜR...BIS...NÄCHSTE. Wenn wir nun noch die Zwischenräume füllen, wird die Bedeutung klarer. Nehmen wir Zeile 10.

```
10 FOR A=1 TO 5
```

Diese Zeile lautet in deutsch FÜR A GLEICH 1 BIS 5, also setze für die Variable A die Werte 1 bis 5 und fange mit 1 an. In Zeile 30 steht dann:

```
30 NEXT
```

in deutsch also NÄCHSTEN Wert. Der C 64 geht hier zum nächsthöheren Wert. Das macht er in Einerschritten, und zwar so lange, bis die letzte Zahl erreicht ist. In unserem Fall ist es die 5. Das muß aber nicht unbedingt bedeuten, daß der C 64 die Schleife fünfmal »durchfliegt«. Der Befehl STEP definiert, in welcher Schrittgröße er die betreffende Variable erhöhen soll. Ändert einmal die Zeile 10 so:

```
10 FOR A=1 TO 5 STEP 2
```

Der C 64 schreibt nur dreimal »ALLES PALETTI FREUNDE«! STEP heißt übersetzt Schritt. STEP 2 bedeutet folglich: »Erhöhe in Zweierschritten«. Statt 1, 2, 3, 4, 5 zählt er nun 1, 3, 5. Wenn er die Zahl 5 erreicht, hat er die Schleife nur dreimal wiederholt. Zugegeben, für dieses Programm ist es recht sinnlos, von 1 bis 5 in Zweierschritten zu zählen. Folgendes Programm macht die Funktion des STEP-Befehls deutlicher. Stellt Euch vor, Ihr braucht für ein Programm nur gerade Zahlen. Überlegt einmal, wie es ohne STEP ginge.

Na, eine Lösung gefunden? Wahrscheinlich nicht, und wenn, dann eine lange und komplizierte. Mit STEP sieht die Lösung folgendermaßen aus:

```
10 FOR Z=0 TO 100 STEP 2
20 PRINT Z;
30 NEXT
```

Drei kurze Programmzeilen und das Problem ist gegessen. Aber der FOR...NEXT-Befehl ist noch genialer. In unserem Programm schreibt der C 64 ziemlich schnell alle Zahlen auf den Bildschirm. Diesen Prozeß können wir mit einer Warteschleife verlangsamen. Ändert im Programm die Zeile 30 in:

```
30 NEXT Z
```

und addiert die Zeile:

```
25 FOR T=1 TO 100: NEXT T
```

Startet nun das Programm mit RUN. Die Zahlen folgen wesentlich langsamer aufeinander. Der Trick ist, daß wir den C 64 in Zeile 25 von 1 bis 100 zählen lassen. Dazu benötigt er ungefähr eine Sekunde. Daher die Pausen zwischen den erscheinenden Zahlen auf dem Bildschirm.

Vielleicht ist Euch aufgefallen, daß hinter jedem NEXT eine Variable steht. In Zeile 25 ist es T, in Zeile 30 Z. Wir können hinter dem NEXT genau definieren, auf welche Schleife sich dieser Befehl bezieht. Theoretisch braucht der C 64 das nicht, denn er bezieht das erste NEXT immer auf die zuletzt geöffnete Schleife. Aber für unsere Übersicht ist es sehr nützlich.

Das Programmieren von Schleifen in Schleifen nennt man Verschachtelung. Dieser Begriff hat eine große Bedeutung. Stellt Euch Schachteln vor. In eine große Schachtel könnt Ihr nur eine kleinere Schachtel stellen. Und in diese paßt wiederum nur eine kleinere. Eine größere Schachtel würde die letztere oder sich selbst zerstören, wenn man sie mit Gewalt hineinstecken wollte. Genauso ist es mit der Verschachtelung in Programmen. Eine Schleife in einer Schleife sollte also nicht über die erste, die Hauptschleife hinausragen wie in folgendem Beispiel:

```
10 FOR Z=0 TO 100
20 PRINT Z
30 FOR A=1 TO 2
40 NEXT Z
50 NEXT A
```

Mein Ziel ist, daß der C 64 zweimal von 0 bis 100 zählt. Daher habe ich die Schleife in Zeile 30 und 50 eingebaut. Nach dem Start mit RUN verläuft zunächst alles wie gewohnt, der C 64 zählt in Zweierschritten von 0 bis 100, jedoch nur einmal. Danach meldet er sich mit »?NEXT WITHOUT FOR ERROR IN 50«. Meine zweite Schleife ragt aus der ersten, der Hauptschleife heraus. Und das mag der C 64 gar nicht. Gehen wir das Ganze noch einmal durch. Ich will, daß der C 64

1. zweimal zählt und
2. jedesmal in Zweierschritten von 0 bis 100 zählt.

In dieser Reihenfolge muß auch verschachtelt werden. So ist es richtig:

```
10 FOR A=1 TO 2
20 FOR Z=0 TO 100
30 PRINT Z
40 NEXT Z
50 NEXT A
```

Weil wir ganz mutig sind, werden wir eine dritte Schleife einbauen, eine Warteschleife. Ergänzt das Programm mit folgender Zeile:

```
35 FOR X=1 TO 200:NEXT X
```

Der C 64 zählt jetzt zweimal von 0 bis hundert, wobei er zwischen jedem Ausdruck einer Zahl von 1 bis 200 zählt, also eine Zeit wartet. Ganz schön kompliziert, oder? Genau das denkt sich der C 64 auch. Wenn zuviele Schleifen ineinander verschachtelt werden, streikt er ganz einfach und beschwert sich mit »?OUT OF MEMORY ERROR«. Bei der Verschachtelung daher zwei Dinge beachten:

1. Immer so wenig wie möglich verschachteln und
2. eine Schachtel nie aus einer anderen herausragen lassen.

Es geht auch anders herum

Bisher haben wir in einer FOR...NEXT-Schleife hochgezählt, zum Beispiel von 1 bis 100 oder 1 bis 5. Wir können aber auch runterzählen:

```
10 FOR A=10 TO 1
20 PRINT A
30 NEXT A
```

Und auch der STEP-Befehl kann umgekehrt verwendet werden:

```
10 FOR A=10 TO 1 STEP -2
20 PRINT A
30 NEXT A
```

Zufrieden lehne ich mich zurück und gehe in Gedanken noch einmal alles durch. Das »Denken in Computer-Bahnen« ist etwas ungewöhnlich, macht mir aber großen Spaß. Wißt Ihr was? Wir beherrschen bereits einen Teil der Programmiersprache Basic! IF...THEN und FOR...NEXT...STEP zum Beispiel gehören alle zum Wortschatz, den jeder Computer-Fachmann beherrschen muß! Wir haben den Fuß in der Tür, in den nächsten Kapiteln stoßen wir sie richtig auf.

Das haben wir gelernt

Variable: Ein Platzhalter für Zahlen, Buchstaben oder Zeichenketten. Es gibt drei verschiedene Typen von Variablen.

ASCII-Code: Ein definierter Zeichensatz, der durch Zahlen darstellbar ist. Der Buchstabe A hat z.B. den ASCII-Wert 65.

CHR$(X): Gibt das Zeichen des ASCII-Codes X aus.

ASC(X$): Gibt den ASCII-Code des Zeichens X$ aus. ASC$(B) ergibt die Zahl 66.

IF...THEN: Stellt an den C 64 eine »Wenn...dann-Bedingung«, z.B. IF A=1 THEN PRINT A. Ist die Bedingung erfüllt, führt er den Befehl hinter THEN aus. Andernfalls springt er zur nächsten Befehlszeile.

FOR...NEXT: Dieser Befehl sorgt dafür, daß ein Programm-Abschnitt mehrmals durchlaufen wird. FOR A=1 TO 5 läßt einen Abschnitt fünfmal durchlaufen.

STEP: Definiert, in welcher Schrittgröße in einer FOR...NEXT-Schleife gezählt werden soll. Bei STEP.2 wird zum Beispiel in Zweierschritten gezählt.

Kapitel 6

Wir bestimmen den Ablauf von Programmen

Wir treffen einen ganz miesen Typen, der uns häufiger begegnen wird. Ein böser Miesling will uns die Arbeit erschweren, aber wir lassen uns weder verunsichern, noch vom Programmieren abbringen.

Bild 6.1: Ein mieser Typ: der Fehlerteufel.

Habt Ihr Euch schon einmal gefragt, was im Inneren unseres Computers passiert? Wir starten einen kleinen Ausflug. Ich will vorher einige Gesichter vorstellen. Bild 6.1 müssen wir uns genau merken. Der Knilch hat immer miese Laune: der Fehlerteufel. Überall hat er seine Finger im Spiel.

Der freundliche Junge in Bild 6.2 heißt Basic-Interpreter. Ihn stelle ich als ersten genauer vor, denn was bedeutet der vielverwendete Begriff »Basic« eigentlich genau?

Bild 6.2: Der Basic-Interpreter übersetzt Basic-Befehle.

Ein Computer ist eine Maschine, die ihre eigene Sprache spricht. Sie arbeitet, wie wir wissen, mit Strom. Dieser wird im Computer durch ein gigantisches System von kleinen Stromleitungen geschickt.

Manche Leitungen sind so klein, daß sie mit bloßem Auge nicht zu erkennen sind. Sie befinden sich in Gehäusen aus Keramik (Bild 6.3). Solche Teile nennt man ICs.

Bild 6.3: Ein IC.

Stellt Euch einen Bahnhof vor (Bild 6.4). Ein Zug fährt auf ihn zu und soll Bahnsteig C anfahren. Eine Weiche lenkt ihn nach rechts zu den Bahnsteigen D und C. Eine weitere Weiche läßt ihn links zum Bahnsteig C abbiegen. Stände die erste Weiche auf links, hätte er sein Ziel nicht erreichen können. Die Stellung der Weichen bestimmt also den Weg des Zuges.

Nichts anderes passiert in einem IC. Die Schienen und Weichen sind die Schaltkreise aus Stromleitungen und der Zug ist der Strom. Abhängig davon, wie der Strom geleitet wird, führt der C 64 Funktionen aus, zum Beispiel »PRINT A«. Er kann eigentlich nur erkennen, in welcher Stromleitung Strom fließt und in welcher nicht.

Dabei unterscheidet er zwischen zwei Zuständen – »Strom an« oder »Strom aus«. Wir stellen diese Zustände mit Zahlen dar. Eine 1 bedeutet »Strom an«, eine 0 »Strom aus«.

START

A B C D

ZIEL

Bild 6.4: Der Strombahnhof.

Daraus können wir eine Sprache entwickeln, die nur aus Nullen und Einsen besteht, sie nennt sich »Maschinensprache« und ist sehr kompliziert. Wir müßten mit Befehlen wie »11010010« oder »10001011« arbeiten.

Tatsächlich programmierten die Pioniere der Computer in dieser Weise. Wir müssen das nicht, denn es gibt »Basic«.

Vielbenutzte Abkürzung

Basic ist eine Abkürzung für »Beginners All Purpose Symbolic Instruction Code« und heißt soviel wie: symbolischer Allzweck-Befehls-Code für Anfänger.

Diese Sprache wurde im Jahre 1965 ursprünglich als Programmierhilfe im technisch-naturwissenschaftlichen Bereich entwickelt, denn sie ist viel einfacher als die Maschinensprache. Heute wird sie vorwiegend für Heimcomputer verwendet.

Im Innern des Computers werden in Basic eingegebene Befehle vom Basic-Interpreter übersetzt (interpretiert) und weitergegeben. Wir sprechen also mit dem C 64 in einer Sprache, die er nicht versteht.

Denkt Euch, Ihr müßtet Euch mit einem Chinesen unterhalten, aber weder spricht er deutsch, noch sprecht Ihr chinesisch. Ein Dritter muß her, der für Euch übersetzen kann – ein Dolmetscher.

Im C 64 steckt ein Dolmetscher, der die Basic-Befehle in Maschinensprache übersetzt. Er heißt Interpreter. Ein Beispiel: Ich setze mich an meinen Computer und gebe ein kurzes Basic-Programm ein.

Die eingegebenen Befehle werden von dem auf Bild 6.2 vorgestellten jungen Herrn in Windeseile in die Maschinensprache übersetzt und weitergeleitet (Bild 6.5).

Der Computer nimmt nun die Arbeit auf, da er die Befehle alle versteht.

Bild 6.5: Befehle durchlaufen verschiedene Stationen.

Das werden wir gleich ausprobieren und weiterführen mit einem bekannten Programm:

```
10 PRINT "HENNING"
20 PRINT "MACHT WEITER"
RUN
```

Auf dem Bildschirm erscheint wie erwartet:

```
HENNING
MACHT WEITER
```

Das ist für uns nichts Neues. Doch leider sehen wir nur das Ergebnis unseres Programms auf dem Bildschirm. Was aber tun, wenn ich noch einmal sehen will, wie es dazu kam? Basic hält dazu den Befehl LIST bereit. Er ruft ein im Speicher abgelegtes Basic-Programm vollständig auf den Bildschirm: Das Programm wird aufgeLISTet. Ein Beispiel erläutert die Arbeitsmöglichkeiten von LIST. Aufgabe: Zeile 10 ändern, statt HENNING soll EGON geschrieben werden. Ich rufe mit LIST RETURN das vollständige Programm auf den Bildschirm und gebe anschließend ein:

```
10 PRINT "EGON"    <RETURN>
```

Der C 64 wirft die alte Zeile aus dem Speicher und ersetzt sie durch die neue. Sicher, das macht er auch ohne LIST. Unser Programm enthält nur zwei Befehlszeilen, da ist es nicht schwer zu wissen, was geändert werden soll. Bei etwas größeren Programmen hat man nicht mehr alle Zeilen im Kopf. LIST ist bei der Fehlersuche in Programmen unentbehrlich. Folgendes Programm enthält absichtlich einen Fehler, tippt es deshalb genauso ab, wie es dort steht.

```
NEW
5 PRINT <SHIFT + CLR/HOME>
10 FOR A=1 T 20 STEP 2
20 PRINT A
30 NEXT A
```

Nach dem Starten mit RUN überrascht der C 64 mit der Fehlermeldung »? SYNTAX ERROR IN 10«, Fehler im Satzbau oder Schreibfehler. Da hat er ganz recht, aber wo liegt er? Drei Fehler wären möglich:

1. Der Befehl FOR ist falsch geschrieben, zum Beispiel FO oder FR, es wurde einfach ein Buchstabe vergessen
2. Der Befehl TO ist falsch geschrieben, zum Beispiel TP oder nur der Buchstabe T.
3. Der Befehl STEP ist nicht richtig geschrieben, zum Beispiel STP oder SSTEP.

Fehler, die in der Praxis oft unterlaufen – Tippfehler. Das Programm steht nicht mehr vor unseren Augen, wir können es nicht überprüfen, wenigstens noch nicht. Es gibt ja LIST, mit dem wir es wieder erscheinen lassen können.

```
LIST  <RETURN>
```

bringt es zum Vorschein. Nun sehen wir, daß uns Fehler Nummer 2 unterlaufen ist und können die nötige Korrektur vornehmen.

Der C 64 meldete einen Fehler in Zeile 10. Da liegt doch die Vermutung nahe, daß Befehlszeilen einzeln LISTbar sind. Geben wir ein:

```
LIST 10 <RETURN>
```

Es funktioniert. Wir müssen also nicht immer das ganze Programm listen. Aber es wird noch toller. Erweitern wir unser Programm um folgende Zeilen:

```
40 PRINT"WIR MACHEN"
50 PRINT"EINEN TEST"
60 END
```

LIST bringt auch ein Programm nur teilweise auf den Bildschirm:

```
LIST -40
```

Heißt nicht, liste die Zeile minus 40, sondern liste bis Zeile 40.

```
LIST 20-
```

Läßt den C 64 das ganze Programm ab Zeile 20 auflisten.

```
LIST 20-40
```

Listet die Zeilen 20 bis 40. Wir können beliebige Abschnitte des Programms listen. Ein Tip noch: die Taste [RUN/STOP] unterbricht den LIST-Vorgang, bei gedrückter CTRL-Taste wird er verlangsamt.

»Brauche INPUT«

Da wir nun eine hilfreiche Stütze zur Fehlersuche haben, wagen wir uns an weitere Befehle. Mit INPUT kann ich einer Variablen beliebige Werte zuweisen und so den Fortgang des Programms durch Eingabe von Zahlen und Buchstaben beeinflussen. INPUT kann übersetzt werden mit »Hineinlegen« oder bissig gesagt »Füttern«. Stößt der C 64 auf diesen Befehl, will er mit Informationen gefüttert werden. Diese Daten legt er in einer Variablen ab, die wir vorher selbst definieren. Ich probiere folgendes Programm aus:

```
10 INPUT A
20 PRINT A
30 PRINT A*3
40 PRINT A+3
50 END
```

Das unscheinbare Programm hat es in sich. Ein Fragezeichen taucht auf. Hat mich der Fehlerteufel erwischt? Nein, das Fragezeichen wird vom Befehl INPUT erzeugt und fordert eine Eingabe von mir. Ich gebe die Zahl 2 [RETURN] ein. Sofort reagiert der C 64. Er schreibt:

```
?2
2
6
5
```

Neues vom Computer

INPUT heißt soviel wie »Eingabe«. Der Computer wartet auf eine Information, die bei jedem Durchlauf des Programms neu festgelegt wird. In Zeile 10 öffnet der C 64 eine Variablen-Schublade und nennt sie A. Den Inhalt fordert er durch das Fragezeichen. Ich wählte die Zahl 2. Für den C 64 steht fest, die Variable A hat den Wert 2. Jetzt kann er in die nächste Befehlszeile springen und das Programm ausführen. Alle A

werden mit 2 ersetzt und ausgerechnet. Das Sternchen »*« ist das Malzeichen (Multiplikation). Zeile 50 beendet das Programm (END = Ende).

Die freie Wählbarkeit des Variablen-Inhalts ist der große Vorteil des INPUT-Befehls. Bei jedem Durchlauf des Programms kann ich einen neuen Wert für A nehmen: 2, 765, 1000000.... Probiert es aus, indem Ihr das Programm immer wieder neu startet. Das Programm steht nicht für eine einzelne Aufgabe, sondern eine bestimmte Aufgabenart.

Bei solchen Programmen ist es sinnvoll, dem Menschen vor dem C 64 zu sagen, was er eingeben soll. In unserem Fall soll er eine Zahl eingeben. Gebt statt einer Zahl einen Buchstaben ein. Der C 64 beschwert sich mit einem »?REDO FROM START«, probier's nochmal. Ganz klar, in Zeile 10 steht

```
10 INPUT A
```

Die Variable A ist Platzhalter für eine Fließkommazahl, eine Zahl, die Stellen hinter einem Komma haben darf. Der C 64 erwartet eine entsprechende Eingabe. Servieren wir ihm einen Buchstaben, lehnt er das glatt mit einer Fehlermeldung ab. Dem Benutzer muß demnach mitgeteilt werden, was er eingeben soll. Die Zeile:

```
5 PRINT"BITTE EINE ZAHL EINGEBEN"
```

macht den Programm-Ablauf nicht nur übersichtlicher, sondern gibt dem Anwender am Computer eine exakte Anweisung.

Wenn wir bestimmen können, daß eine Zahl in den C 64 »gefüttert« werden soll, können wir auch die Eingabe eines Buchstabens oder ganzer Wörter definieren. Statt Fließkommazahlen bedienen wir uns der String-Variablen. Zur Erinnerung: Eine String-Variable ist eine Zeichenkette wie zum Beispiel »KARL« oder »FRIDOLIN«.

```
10 PRINT "HALLO! ICH BIN DEIN COMPUTER"
20 PRINT "WER BIST DU";
30 INPUT DU$
40 PRINT "GUTEN TAG ";DU$
50 PRINT "WENN DU WEITER"
60 PRINT "HENNING PACKT AUS LIEST"
70 PRINT "WERDEN WIR GUTE FREUNDE"
80 PRINT DU$;" WIR SCHAFFEN DAS SCHON"
90 END
```

Gestartet wird wieder mit RUN. Leute, erinnert Ihr Euch an den schrägen Vogel in Bild 6.1? Der alte Miesmacher stellt mir ein Bein nach dem anderen. Ich muß jedes Zeichen genau abtippen, jedes Freizeichen beachten und Stück für Stück den Störenfried aus meinem schönen Programm zerren. Der LIST-Befehl hilft dabei sehr. Mit RUN läuft alles nach Plan. Das Ergebnis sieht dann auf dem Bildschirm so aus:

```
RUN
HALLO! ICH BIN DEIN COMPUTER
WER BIST DU? HENNING
GUTEN TAG HENNING
WENN DU WEITER
HENNING PACKT AUS LIEST
WERDEN WIR GUTE FREUNDE
HENNING WIR SCHAFFEN DAS SCHON
READY
```

Schritt für Schritt gehen wir nun durch dieses fantastische Programm: Zeilen 10 und 20 lassen den Computer die angegebenen Wörter auf den Bildschirm drucken. In Zeile 30 geht es richtig los. Der C 64 fordert mit einem Fragezeichen eine Eingabe: meinen Vornamen. Die dazugehörige Variable nennt sich DU$. Die Aufschrift des darinliegenden Zettels ist »HENNING«. Durch meine Eingabe »HENNING« habe ich die Antwort auf die Frage des Computers gegeben.

```
WER BIST DU? HENNING
```

Die Zeile 40 gibt dem Rechner zwei Anweisungen hintereinander:

1. Schreibe »GUTEN TAG« (Achtung: Zwischen dem G von TAG und dem Anführungszeichen ist eine Leerstelle, die muß bleiben, sonst schreibt er »GUTEN TAGHENNING«!)

2. Schreibe dahinter den Inhalt der Variablen DU$ (HENNING)

Aus den zwei Teilen entsteht »GUTEN TAG HENNING«.

Zeilen 50 und 70 schreiben den Inhalt der PRINT-Anweisung auf den Bildschirm. In Zeile 80 wird der Inhalt der String-Variablen DU$ vor »WIR SCHAFFEN DAS SCHON« geschrieben:

```
HENNING WIR SCHAFFEN DAS SCHON
```

Zeile 90 beendet das Programm. Das macht richtig Spaß, der Computer scheint lebendig zu werden. Zum Abschluß ein kleiner Tip: Probiert unser Namen-Programm ohne die Semikolons in Zeile 20 und 40. Welche Veränderung stellt Ihr fest und was gefällt Euch besser? Ich mag das Programm mit Semikolons lieber.

Die Alternative – GET

Der Befehl GET hat in Programmen eine ähnliche Funktion wie INPUT. Er heißt auf deutsch »hole« oder »bekomme«. Der C 64 holt sich durch diesen Befehl ein Zeichen, welches über die Tastatur eingegeben wird, ohne den Ablauf des Programms zu unterbrechen. Bevor wir uns genauer mit diesem Befehl beschäftigen, müssen wir uns mit

der Arbeitsweise der Tastatur vertraut machen. Zu diesem Zweck geben wir ein kleines Programm ein:

```
NEW
10 FOR I=1 TO 5000
20 NEXT I
RUN
```

Während das Programm läuft (Dauer etwa 5 Sekunden), drücken wir auf einige Tasten, zum Beispiel die K- und die L-Taste. Zunächst tut sich überhaupt nichts, der C 64 macht seelenruhig sein Programm weiter. Doch nach Ablauf des Programms meldet sich der C 64 nicht nur mit READY, sondern mit

```
READY
KKKKKLLLLL
```

Auf dem Bildschirm erscheinen fünf K und fünf L, weil ich fünfmal die K- und fünfmal die L-Taste gedrückt habe. Warum? Der Tastaturpuffer macht es möglich. Beim Abarbeiten der beiden Programmzeilen schaut der C 64 sechzigmal pro Sekunde nach, ob wir zwischendurch eine Taste gedrückt haben. Diese eingegebenen Buchstaben stören den Ablauf des Programms nicht. Sie werden am Ende aus dem Tastaturpuffer geholt und ausgedruckt. Der Tastaturpuffer ist ein kleiner Speicher, der nur für die Tastatur zuständig ist. Die eingegebenen Buchstaben geraten nicht in Vergessenheit. Im Puffer ist Platz für 10 Zeichen. Der GET-Befehl liest Variablen aus diesem Tastaturpuffer.

Machen wir uns die Sache im Unterschied zum INPUT-Befehl klar. Wir haben in unserem Programm gemerkt, der C 64 stoppt bei INPUT den Ablauf des Programms und wartet mit einem Fragezeichen auf eine bestimmte Eingabe. Das macht GET nicht. Nehmen wir an, wir sind Computer und treffen im Verlauf eines Programms auf eine GET-Anweisung. In diesem Fall schauen wir in Windeseile im Tastaturpuffer nach, ob ein Zeichen eingegeben wurde. Ist dies der Fall, nehmen wir es auf den Rücken und düsen unverzüglich weiter. Tja, das ist alles etwas hochgestochen, oder? Nehmen wir ein Beispiel:

```
10 PRINT "ANFANG"
100 PRINT "NOCH EINMAL (J/N)?"
110 GET E$:IF E$="" THEN 110
120 IF E$="J" THEN 10
130 IF E$ <> "N" THEN 100
140 PRINT "DU WILLST NICHT MEHR? DANN TSCHUESS"
RUN
```

Hier ein Probelauf des Programms:

```
ANFANG
NOCH EINMAL (J/N)?
ANFANG
NOCH EINMAL (J/N)?
DU WILLST NICHT MEHR? DANN TSCHUESS
```

In Zeile 10 wird einfach ANFANG auf den Bildschirm geschrieben. Zeile 100 druckt die angegebenen Worte aus. Zeile 110 ist sehr interessant. Hier bekommt der C 64 folgenden Befehl: »Hole Dir die Variable E$. Schaue im Tastaturpuffer nach, ob eine Eingabe gemacht worden ist und weise sie der Variablen E$ zu. Wenn der Puffer leer ist, also "" enthält, dann bleibe in Zeile 110.« Auf diese Weise bleibt der Ablauf des Programms in Zeile 110 stecken, bis wir ein Zeichen eingeben und es damit in den Tastaturpuffer legen.

Jetzt kann der C 64 weitermachen, da der Puffer nicht mehr leer ist. GET liest den Inhalt des Puffers, nimmt das eingegebene Zeichen unter der Bezeichnung »E$«-Variable auf den Rücken und heizt weiter. In 120 wird nun der Inhalt von E$ überprüft. Wenn E$ den Buchstaben »J« enthält, so springt das Programm zurück in Zeile 10. Bei allen Zeichen UNGLEICH »N«, wird in Zeile 100 weitergemacht, da die Frage falsch beantwortet wurde. Es sollte entweder »J« oder »N« gedrückt werden. Die eckigen Klammern in Zeile 130 bedeuten »ungleich«. Nehmen wir an, wir geben nach der Frage »NOCH EINMAL (J/N)?« ein »N« ein. In diesem Falle übergeht der Computer die Zeilen 120 und 130, da die Bedingungen nicht erfüllt sind. Er trifft auf Zeile 140, druckt den Inhalt »DU WILLST NICHT MEHR? DANN TSCHUESS« aus und beendet das Programm. Ich gebe zu, die Sache ist nicht ganz einfach. Meine grauen Gehirnzellen haben sich eine ganze Weile gegen die Denkart eines Computers gewehrt, weil sie etwas ungewohnt ist. Ihr fragt Euch bestimmt: Was soll das überhaupt?

Stellt Euch vor, unser kleines Beispiel-Programm ist in ein größeres eingebunden (die Zeilen 10 bis 100 bieten sich hier an). Schon ist der Sinn des Listings klar ersichtlich! Am Ende fragt das Programm: Willst Du das Programm noch einmal ganz von vorne beginnen, dann gehe zum Anfang zurück (Zeile 10), ansonsten beende das Programm. Wenn Ihr das Programm noch einmal benutzen wollt, müßt Ihr nur »J« eingeben und die Sache startet. Das ist eine deutliche Arbeitsersparnis. Faulheit ist Trumpf, nur die Dummen arbeiten mehr als nötig!

Das, was wir eben gemacht haben, nennt sich »menügesteuert«. Wir haben in unserem letzten Beispielprogramm mit »J« oder »N« das Programm gesteuert! Der Computer hat uns die Wahlmöglichkeiten praktisch in die Hand gelegt. Wir mußten nur lesen, was eingegeben werden soll, entscheiden und dann weitermachen. In großen Programmen kommt diese Arbeits- und Lenktechnik häufig vor, weil wir uns damit prima zurechtfinden können.

```
10 PRINT"SIE KOENNEN NUN DIE RAHMENFARBE AENDERN"
20 PRINT"R = ROT"
30 PRINT"G = GRUEN"
40 PRINT"S = SCHWARZ"
50 PRINT"E = ENDE"
60 GET A$:IF A$="" THEN 60
70 IF A$="R" THEN POKE 53280,2
80 IF A$="G" THEN POKE 53280,5
90 IF A$="S" THEN POKE 53280,0
100 IF A$="E" THEN END
110 GOTO 60
```

Durch POKE 53280,X können wir die Rahmenfarbe ändern. Zerbrecht Euch nicht den Kopf über diesen Befehl. Wir werden später noch mal genauer auf ihn stoßen. Dieses Programm ist lediglich eine Demonstration der Menüsteuerung. So, jetzt werden wir uns auch mal »menüsteuern«: Wer Lust hat, eine Weile zu entspannen und unser Spiel »The Great Giana Sisters« spielen will, sagt »Ja« und legt die Diskette ins Laufwerk, ich kann den Pausenwunsch gut verstehen. Die ganz Eisenharten sagen »Nein« und blättern ins nächste Kapitel um.

Das haben wir gelernt

Basic: Eine einfache Programmiersprache, die das Arbeiten mit dem Computer vereinfacht.

Interpreter: Er übersetzt unsere Basic-Befehle in die für den C 64 verständliche Maschinensprache.

LIST: Listet ein im Speicher befindliches Basic-Programm ganz oder teilweise auf den Bildschirm.

INPUT: Stößt der C 64 im Verlauf eines Programms auf diesen Befehl, verlangt er die Eingabe von Daten. Dies macht er mit einem Fragezeichen kenntlich.

GET: Fragt die Tastatur ab und erkennt, ob eine Taste, beziehungsweise welche Taste gedrückt wurde.

Kapitel 7

Programmieren mit Planung

Die Zeiten des Selbstprogrammierens beginnen! »Strukturiertes Programmieren« heißt das Schlagwort. Eine übersichtliche Programmiertechnik, die eine Idee Schritt für Schritt in ein perfektes Listing verwandelt. Das Ganze ist so einfach wie Eisschlecken.

Ein Basic-Programm zu schreiben ist gar nicht so schwer. Werkzeuge sind unsere Ideen und die verschiedenen Basic-Befehle wie INPUT, GOTO und IF...THEN. Die Grundlagen haben wir in den vorhergegangenen Kapiteln kennengelernt, jetzt werden sie richtig angewendet. Im Zusammenspiel von Ideen, Grundkenntnissen und dem »strukturierten Programmieren« entsteht ein Riesenprogramm von 20 Zeilen. Also hinein ins Vergnügen.

Ziel erkannt

Viele Programmierer tippen sich unter Haareraufen und Nägelkauen die Finger wund, weil ihr sauer erarbeitetes Programm nicht funktioniert. Wir nicht! Das »strukturierte Programmieren« schließt Flüchtigkeitsfehler weitgehend aus: In fünf Schritten formt sich eine Idee zu einem vollständigen Programm. Es wird nicht sofort mit dem Programmieren begonnen! Der Verlauf und die Funktion des neuen Listings wird vor dem Arbeiten am Computer »auf dem Trockenen« festgelegt, ohne einen einzigen Basic-Befehl. Die fünf Schritte sind folgende:

1. Es wird ein »Algorithmus« entworfen: Im allgemeinen Sprachgebrauch steht der Begriff für eine bestimmte Rechenart, für uns bedeutet er eine bestimmte Denkart. Wir überlegen, welche Schritte in welcher Reihenfolge das neue Programm zum Ziel führen. Diese Schrittfolge schreiben wir peinlich genau nieder, als ganze Sätze.
2. Wir zeichnen den Algorithmus. Jeder einzelne Befehlsschritt wird in ein besonderes Kästchen gesetzt, Pfeile kennzeichnen den Programmablauf und markieren Schlei-

fen. Die verwendeten Kästchen stellen die Arbeitsvorgänge bildlich dar, der Programmablauf wird »vorgezeichnet«. Ein solches Schaubild wird »Flußdiagramm« genannt (Bild 7.1).

3. Jetzt erst verwandeln wir das Flußdiagramm in Basic-Befehle (codieren)!
4. Das Programm wird auf Programmierfehler getestet, »Fehlersuche«.
5. Wir »dokumentieren« unser Programm. Es werden Bemerkungen in das Programm eingefügt. Sie verändern den Ablauf des Listings nicht und erleichtern uns ein erneutes Einsteigen in die verschiedenen Befehle (zum Beispiel wenn wir uns in zwei Wochen wieder mit dem Programm beschäftigen wollen).

Das da oben klingt alles sehr fremd. Auf alle Fälle: Wir werden jetzt ein langes Programm schreiben. Bevor wir uns dieser Aufgabe zuwenden, klären wir an einem kühlen und praktischen Beispiel die Begriffe »Algorithmus« und »Flußdiagramm«.

Eis kaufen und Programme schreiben

Stellt Euch vor, Ihr bekommt bei einer Affenhitze mitten im Sommer riesige Lust auf ein Monstereis. Wer jetzt einfach zum Eismann rennt und sich ein Super-Giganto-Eis bestellt, kann genauso auf die Nase fallen wie ein planloser Programmierer. Kein Geld dabei, das gibt Ärger!

Der Eiskauf muß geplant werden, genauso wie das Programmieren. Wir erstellen einen Algorithmus für das Programm »Eiskaufen«:

1. Affenhitze, der Eismann kommt, das Wasser läuft im Mund zusammen.
2. In die Tasche greifen, ist Geld da?
3. Vorsichtig über die Straße laufen.
4. Das Eis bestellen.
5. Das Eis entgegennehmen, essen.
6. Ein zufriedenes Gesicht aufsetzen.
7. Bei Bedarf: GOTO 4.

Der Algorithmus ist der Lösungsweg zum perfekten Programm. Im Flußdiagramm sieht das Ganze so aus, wie es in Bild 7.1 dargestellt ist.

Start
Eismann kommt
Du willst ein Eis
Geld?
nein
Geld holen
ja
Vorsichtig über die Straße gehen
Eis kaufen
(Computer)–Eismann gibt Eis
Eis essen
zufriedenes Gesicht
noch ein Eis?
ja
nein
Ende

Bild 7.1: Das Eiskauf-Flußdiagramm.

Vielsagende Rechtecke

In einem Flußdiagramm hat jedes Symbol eine andere Bedeutung. Es gibt: Ovale Zeichen (Start, Ende), Rechtecke, Rauten und verbindende Linien (Bild 7.2).

Interner Programmschritt (z.B. Rechnen)
Verzweigung, Test– oder Entscheidungssymbol
Ein/Ausgabe Symbol
Zusammenführung
Grenzstelle (Start/Ende)
Ablauflinie

Bild 7.2: Diese Zeichen benötigen wir beim strukturierten Programmieren.

Die ovalen Symbole stehen für Anfang und Ende eines Programms. Mit ihnen machen wir lediglich klar, an dieser Stelle beginnt das Programm und hier hört es auf. Ein Rechteck bezeichnet alle Programmschritte, in denen der Computer rechnet oder er direkte Tätigkeiten ausführt, als »internen« Programmschritt (intern=innerhalb des Computers). Wir sehen bei solchen Schritten meistens nicht, daß der C 64 arbeitet. Soll er zum Beispiel die Zahl 12 durch 3 teilen, so macht er das im Verborgenen, er rechnet intern. Eine Raute ist ein »verbogenes« Rechteck. Sie bedeutet eine Ein- oder Ausgabe. Hier müssen wir dem Computer etwas mitteilen (INPUT), oder der Computer druckt zum Beispiel eine Information aus (PRINT). Das dritte viereckige Zeichen in Bild 7.2 ist eine »Kreuzung«. Der Computer kann links oder rechts »abbiegen«. Hier verlangt der C 64 von uns eine Entscheidung, zum Beispiel ob wir das Programm beenden oder nochmal sehen wollen.

Fahren wir das Flußdiagramm einmal gedanklich ab. Das runde Zeichen am Anfang kennzeichnet den Start. Im ersten Rechteck arbeitet der Computer: Der Eismann (in unserem Falle der Computer) kommt. Das zweite Zeichen zeigt unsere eigene Aktion, der Computer hat damit wenig zu tun. Wir wollen ein Eis! An dieser Stelle taucht das »Kreuzungs«-Symbol auf, denn es gibt für den weiteren Programmablauf zwei Möglichkeiten. Haben wir Geld oder nicht? Zum Eiskaufen benötigen wir Geld. Wenn wir welches haben (»Ja«), fährt das Programm einfach weiter, bei »Nein« müssen wir abbiegen und erst Geld holen. Anschließend stoßen wir wieder auf den normalen Programmablauf. Jetzt können wir über die Straße gehen und das Eis kaufen. Danach tritt der Computer-Eismann in Aktion: Er gibt uns das Eis, wir essen es und legen ein zufriedenes Gesicht auf. Die letzten beiden sind eindeutig unsere eigenen Handlungen, die Rauten-Zeichen geben das an. Gegen Ende des Programms taucht eine Zwickmühle auf: Wollen wir noch ein Eis? Bei »Nein« ist das Programm sofort zu Ende, bei »Ja« wird eine Schleife zurück zu »Eis-Kaufen« geflogen. Diese Schleife kann beliebig oft durchfahren werden, sobald einmal die Entscheidung »Nein« getroffen wird, ist das Programm beendet.

Wie einfach kann ein Programm mit einem vernünftigen Bild sein. Jeder Schritt ist durch das Flußdiagramm klar und verständlich gegliedert. Wir kennen die Begriffe »Algorithmus« und »Flußdiagramm«. Noch eine kleine Veränderung am Flußdiagramm und dann hindert uns nichts mehr am Programmieren. Die Bedeutungen der einzelnen Symbole sind bekannt und verstanden, jetzt wird gerodelt.

Langsames Heranpirschen

Das erweiterte Flußdiagramm ist etwas schwieriger, weil es mehr Möglichkeiten beinhaltet. Außerdem fließt eine Menge Lebenserfahrung ein: Wer zuviel Eis ißt, bekommt Bauchschmerzen, also Vorsicht. Zuerst gehen wir die Sache im Kopf durch. Wie viele Entscheidungen muß ich treffen und was passiert?

Es gibt drei Entscheidungen nach dem Eintreffen des Eismanns:

1. Will ich überhaupt ein Eis?
2. Habe ich genug Geld dabei?
3. Will ich nach meinem ersten Eis ein weiteres?

Wenn ich mich bei 1. für »N« wie »Nein« entscheide, ist das Programm bereits an dieser Stelle fast vorbei. 2. und 3. setzen meinen Heißhunger auf Eis voraus. Wenn ich akuten Geldmangel feststelle, muß ich Geld holen gehen. Habe ich nach einem Eis nicht genug, kaufe ich mir ein neues. Zu diesem Zweck steige ich von neuem in den Programmabschnitt »Eiskaufen« ein, ich fliege im Programm in einer Schleife ein

Stück zurück. Drei grundlegende Strukturen des »Eis-Programms« liegen fest, beginnen wir mit dem Zeichnen, um die Idee zu realisieren.

Wir nehmen ein großes Blatt, oben wird in einem Oval »Start« geschrieben und ganz unten »Ende«.

Der erste Schritt nach »Start« ist klar: Der Eismann kommt, dies soll der C 64 mit einer Meldung auf dem Bildschirm bestätigen.

Die erste Entscheidung lautet: Eis oder nicht? Hierfür benötigen wir das Entscheidungssymbol, von dem zwei Pfeile abgehen, einer nach unten, neben dem »J« für »Ja« steht und einer nach links mit »N« für »Nein«.

Den Pfeil »Nein« können wir sofort zum Ende des Programms führen, denn wer kein Eis will, kauft auch keins.

In die große Lücke zwischen Anfang und Ende zeichnen wir mit viel Platz zwei Entscheidungssymbole. Im ersten steht »Geld oder nicht?«, im zweiten »Noch ein Eis oder nicht?«.

Bild 7.3 zeigt das Flußdiagramm (am besten verdeckt Ihr die noch unbekannten Teile mit einem Blatt oder Ihr zeichnet es soweit nach, wie wir gekommen sind).

Die wichtigsten Schritte haben wir uns klargemacht, jetzt kommen die Zwischenschritte und Verbindungspfeile. Als Orientierungshilfe habe ich alle Symbole durchnumeriert.

Bevor wir eine Entscheidung bei 5 treffen, müssen wir uns diese Frage erst einmal stellen, das geschieht durch 4. 6 bis 8 sind vom »kleinen« Flußdiagramm bekannt.

Symbol 13 führt uns zwischen 6 und 7, der Eiskauf beginnt von vorne. Wer zuviel Eis ißt, oder es nicht verträgt, bekommt in 10 Bauchweh und hat keine Lust mehr.

1 Start
2 Eismann kommt
3 Ent-scheidung
>>N<< (Nein)
>>J<< (Ja)
4 Hast Du Geld?
5 Geld oder nicht ?
>>N<< (Nein)
>>J<< (Ja)
12 Geld holen
6 Über die Straße gehen Eis kaufen
7 Computer gibt Eis Anzahl der Eis
8 Willst Du noch eins?
9 Noch ein Eis oder nicht?
>>J<< (Ja)
13 Noch ein Eis
>>N<< (Nein)
10 Bauchweh
14 Schade
11 Ende

```
10   PRINT: "<SHIFT/CLR HOME>"
20   PRINT: "DER EISMANN IST DA"
30   PRINT: PRINT "WILLST DU EIN EIS?"
40   INPUT E$
50   IF E$ = "N" THEN PRINT: PRINT "SCHADE" GOTO 200
60   PRINT: PRINT "HAST DU GELD?"
70   INPUT G$
80   IF G$ = "N" THEN PRINT "DU GEHST ERST GELD HOLEN"
90   PRINT: PRINT "DU GEHST VORSICHTIG UEBER DIE STRASSE"
100  PRINT: PRINT "DU KAUFST EIN EIS"
110  X=X+1
120  PRINT: PRINT "WILLST DU NOCH EINS?"
130  INPUT N$
140  IF N$ = "N" THEN 170
150  PRINT: PRINT "DU KAUFST
       EIS NUMMER ";X+1
160  GOTO 110
170  PRINT: PRINT "DU HAST BAUCHSCHMERZEN BEKOMMEN"
180  IF X=1 THEN PRINT: PRINT "EIS BEKOMMT DIR NICHT" GOTO 200
190  PRINT: PRINT X; " EIS BEKOMMT DIR NICHT"
200  END
```

Bild 7.3: Das Flußdiagramm führt uns zu einem vollständigen Programm.

Basic öffne dich!

Jetzt wird codiert, das Flußdiagramm ist fertig. An die Arbeit! Wir müssen uns einige Dinge überlegen:

I. Der Computer soll mir die Programmschritte mitteilen und Fragen an mich stellen.

II. Die Entscheidungsfragen 3, 5 und 9 müssen in sinnvolles Basic übersetzt werden. Ihre Beantwortung darf den Programmablauf nicht durcheinanderbringen.

III. In das Programm muß eine Zählvariable (X) eingebaut werden, die die Anzahl der gemampften Eis angibt.

Problem I löst sich mit dem PRINT-Befehl. Der Computer druckt die Information oder die Aufforderung auf den Bildschirm aus. Die Symbole 2 und 6 sehen in Basic so aus:

```
2. PRINT "DER EISMANN IST DA"
6. PRINT "DU GEHST VORSICHTIG UEBER DIE STRASSE"
   PRINT "DU KAUFST EIN EIS"
```

Die Zählvariable nennen wir X. Die dazugehörige Gleichung ist X=X+1. Diese Gleichung wird in die Eiskauf-Schleife zwischen Symbol 7 und 13 eingebaut. Bei jedem Schleifendurchlauf erhöht sich X um eins und gibt die Menge an Eis an, die wir gegessen haben, wenn hinterher der Wert für X mit PRINT X ausgegeben werden soll.

Die Lösung für Problem III ist der INPUT-Befehl. Der weitere Verlauf des Programms hängt an dem Kreuzungspunkt von der gemachten Eingabe ab. Diese zwei Programmzeilen zeigen den Vorteil von INPUT an. Im Programm benutzen wir es dreimal.

```
70 INPUT E$
80 IF E$="N" THEN 210
```

Zeile 70 besagt folgendes: Gib mir einen Buchstaben für die String-Variable E ein. In Zeile 80 scheiden sich die Wege je nach Eingabe: Ist der Buchstabe »N« eingegeben worden, so gehe direkt zur Programmzeile 210. Bei ALLEN ANDEREN EINGABEN gehe NICHT in Zeile 210, sondern mache im Programmablauf weiter (Zeile 90). Kurz codiert ist halb gewonnen!

Wir kommen zu Potte

Die Codiermaschine läuft an. Wir gehen das Flußdiagramm Schritt für Schritt durch und verwandeln es in Basic-Befehle. Symbol 1 verliert an Bedeutung, dafür wird 2 in vier verschiedene Programmzeilen verwandelt. Sie sind neben Symbol 2 im Flußdiagramm angegeben. Zeile 10 löscht den Bildschirm (Ihr erhaltet das merkwürdige Herz zwischen den Anführungsstrichen durch gleichzeitiges Drücken von [SHIFT] [CLR/HOME]).

Der erste PRINT-Befehl in 30 bewirkt nichts anderes als eine Freizeile. Der Programmausdruck wird übersichtlicher, da die ausgedruckten Zeilen sich nicht so sehr auf den Pelz rücken. Diesen Trick wenden wir mehrmals an, wenn er Euch nicht gefällt, könnt Ihr ihn weglassen. Zeile 40 leitet das Entscheidungssymbol 3 ein. INPUT E$ ruft auf dem Bildschirm ein Fragezeichen hervor, das mit einem Buchstaben beantwortet werden muß. Für »N« wie »Nein« tritt Zeile 50 in Kraft: Schreibe eine Leerzeile, dann »SCHADE« und gehe an das Ende des Programms (Zeile 200). Damit ist der erste lange Programmweg durchlaufen. Bei allen anderen Eingaben als »N« geht der Computer zu Symbol 4 und wir auch. Diesen Programmschritt verwandeln wir in die Zeilen 60 und 70. In 70 arbeiten wir mit der gleichen Technik wie in 40. Wenn wir kein Geld haben, gehen wir welches holen, der C 64 druckt den Text aus Zeile 80 und kehrt dann zurück.

Immer tiefer hinein in die Materie

Die Zeilen 80 und 90 fallen uns nicht schwer, es sind einfache PRINT-Befehle. Schwieriger ist Symbol 7. Dies ist die oben bereits erwähnte Eis-Zähl-Stelle. Bei jedem Eiskauf wird um eins erhöht. Der Computer beginnt mit X=0 und erhält beim ersten Durchlauf X=1. Die Diagrammstation 8 ähnelt im Aufbau Symbol 4: INPUT wird wieder angewendet. Jetzt kommt eine schwierige Stelle: Die Schleife des Eis-Essers. Für alle Angaben außer »N« geht es bei Nummer 13 weiter. Der Computer druckt in Zeile 150 hinter »DU KAUFST EIS NUMMER« den Wert von X+1 aus. Danach geht es in 160 zurück zum Eiskauf in Zeile 110.

Der oben beschriebene Weg läuft bei allen Eingaben außer »N« ab, wenn dies nicht der Fall ist, müssen wir die ganze Schleife überspringen, es geht in Zeile 170 weiter. Symbol 10 vereinigt drei Programmzeilen. Bei unserem Programm bekommen wir immer Bauchschmerzen, egal wieviel Eis wir essen! Gemein, aber wahr. In 180 handeln wir den »Ein-Eis-Fall« ab. Wir haben nur ein einziges Eis gegessen und in Zeile 130 gaben wir »N« ein. Die Zählvariable hat dann den Wert X=1 und stimmt mit Zeile 180 überein. Bei Zeile 190 kommen alle Viel-Eis-Esser zur Geltung: jeder Fall, der über 1 Eis hinausgeht. In den beiden Zeilen 180 und 190 müssen wir uns an die Ergebnisse der Schleife »WILL ICH NOCH EIN EIS?« anpassen, es gibt hier ja mehrere mögliche Ergebnisse. 180 erklärt in einem fehlerfreien Satz: EIS BEKOMMT DIR NICHT und 190 erklärt 5, 10 oder 10000 EIS BEKOMMEN DIR NICHT.

Das war es im Grunde genommen. Zeile Nummer 200 rundet das feine Programm ab. Achtung: Die neben dem Programm (Listing 7.1) abgedruckten Zahlen in Klammern gehören nicht zum Programm und dürfen nicht miteingegeben werden! Es sind die Prüfsummen der Eingabehilfe namens »Checksummer«, der wir uns gegen Ende des Programms zuwenden. Beachtet sie im Moment einfach nicht, ihre Funktion schauen wir uns wie gesagt gleich an!

```
5 REM *** EIS-KAUF-PROGRAMM ***                    <Ø38>
1Ø PRINT "{CLR}"                                   <254>
2Ø PRINT "DER EISMANN IST DA"                      <117>
3Ø PRINT "WILLST DU EIN EIS?"                      <2Ø8>
4Ø INPUT E$                                        <17Ø>
5Ø IF E$="N" THEN PRINT:PRINT"SCHADE":GOTO
   2ØØ                                             <Ø68>
6Ø PRINT:PRINT"HAST DU GELD?"                      <14Ø>
7Ø INPUT G$                                        <216>
8Ø IF G$="N" THEN PRINT"DU GEHST ERST GELD
   HOLEN"                                          <Ø63>
9Ø PRINT:PRINT"DU GEHST VORSICHTIG UEBER D
  IE STRASSE"                                      <176>
1ØØ PRINT:PRINT"DU KAUFST EIN EIS"                 <244>
1Ø5 REM *** JETZT WIRD DAS EIS GEZAHLT ***         <Ø45>
11Ø X=X+1                                          <138>
12Ø PRINT:PRINT"WILLST DU NOCH EINS?"              <Ø96>
13Ø INPUT N$                                       <Ø76>
14Ø IF N$="N" THEN 17Ø                             <1ØØ>
15Ø PRINT:PRINT"DU KAUFST EIS NUMMER ";X+1         <Ø9Ø>
16Ø GOTO 11Ø                                       <1Ø4>
17Ø PRINT:PRINT"DU HAST BAUCHSCMERZEN BEKO
    MMEN"                                          <Ø2Ø>
18Ø IF X=1 THEN PRINT:PRINT"EIS BEKOMMT DI
    R NICHT":GOTO 2ØØ                              <Ø37>
19Ø PRINT:PRINT X;" EIS BEKOMMEN DIR NICHT
    "                                              <Ø96>
2ØØ END                                            <2Ø2>
```


Listing 7.1: Das Ergebnis unserer Programmier-Kunst.

Vorsicht, Fehlerteufel!

Leute, wir sind fast fertig. Jetzt müssen wir das Programm nur noch auf Fehler prüfen. Wir gehen durch und haben Glück, sehr sorgfältig gearbeitet. Das klappt nicht immer so einfach! Besonders die Numerierung der Zeilen fällt häufig schwer. Stellt Euch vor, Ihr habt ein Programm fast fertig, mit allen Schikanen. Plötzlich fällt Euch ein: »Ich habe eine Zeile vergessen!« Sie muß nachträglich eingebaut werden. Das wirft die gesamte Numerierung Eures Programms durcheinander, da sich ab einer gewissen Zeile alles nach hinten verschiebt. In so einem Fall müssen wir aufpassen wie die Schießhunde. Bei unserem Programm gibt es einige schwierige Stellen. Zum Beispiel müssen wir in Zeile 50 genau das Ende des Programms vor Augen haben. In dieser Zeile entscheidet sich der Benutzer für oder gegen ein Eis. Bei »Ja« läuft alles normal, bei »Nein« muß das Ende des Programms vorweggenommen werden. Wie aber weiß ich beim Programmieren schon am Anfang, in welcher Zeile mein Programm aufhört?

Wir dürfen den Überblick nicht verlieren. Deswegen ist es ratsam, einen hohen Wert für die Endzeile anzunehmen und später muß genau kontrolliert werden. Schwierig ist auch der Umgang mit Zeilensprüngen wie in Befehl Nummer 140. Wenn plötzlich nach 140 eine Zeile eingebaut werden muß, stimmt der Befehl THEN 170 nicht mehr und das Programm geht ganz übel baden. Die Neunumerierung von Zeilen muß mit größter Sorgfalt geschehen. Beachtet dabei die Sprünge zwischen den Programmzeilen. Bei einer Neunumerierung müssen auch alle Werte hinter dem Befehl GOTO entsprechend geändert werden, sonst kommt Euer Programm ganz schön durcheinander. Es zeigt sich wieder, wie wertvoll eine von Anfang an gute Planung ist.

Zurück zu unserem Programm, das wir schnell in den Computer eintippen und ausprobieren: es funktioniert! Nach 255 Eis beende ich meinen ersten Programmdurchlauf. Fantastisch, aber diese Bauchschmerzen!

Ein einziger Punkt ist noch offen: die Dokumentation. Wenn ich morgen mein Programm anschaue, weiß ich vielleicht nicht mehr so genau, was die einzelnen Zeilen bewirken. Mit dem Basic-Wort REM können wir jede beliebige Information in unser Programm einbauen, zum Beispiel den Namen oder die Funktion des nächsten Programm-Abschnittes. Viele Programmierer benutzen die ersten Programmzeilen als REM-Zeilen und schreiben dort ihren Namen und ihre Adresse rein oder ähnliche Dinge. Wenn eine Programmzeile mit REM beginnt, so ignoriert sie der C 64. Je mehr REMs ich einbaue, desto besser kann ich die verschiedenen Abläufe am Programm selber erkennen. Hier einige Beispiele:

```
  5 REM **EIS-KAUF-PROGRAMM**
105 REM **JETZT WIRD DAS EIS ZUSAMMENGEZAEHLT**
135 REM **SCHLEIFE ODER NICHT SCHLEIFE**
```

Fügt diese Beispiele in unser Programm ein und schaut es Euch dann an. Denkt Euch vielleicht ein paar REM-Bemerkungen zusätzlich aus. Eines sollte dabei jedoch peinlichst genau beachtet werden. Ein GOTO-Befehl darf niemals auf eine Programmzeile verzweigen, die ausschließlich einen REM-Befehl enthält. Der C 64 arbeitet dann zwar weiter wie gewohnt, doch birgt diese Technik einen entscheidenden Nachteil. REM-Zeilen benötigen wie alle anderen Zeilen auch Speicherplatz. Bei ganz langen Programmen kann es schon mal passieren, daß die letzten paar Befehlszeilen nicht mehr in den Speicher passen, der für Basic-Programme gedacht ist. Was schmeißen wir also als erstes raus? Genau! Die REM-Zeilen, da sie keine Funktion im Programm-Ablauf haben. Löschen wir eine REM-Zeile, auf die ein GOTO verweist, wird das Programm an dieser Stelle abstürzen und uns mit der Fehlermeldung: »?LINE DOES NOT EXIST« (»Diese Zeile gibt es nicht«) überraschen.

Das reicht erst einmal. Die nächste Zeit wird mit Programmieren und Denken ausgefüllt sein. Wir können nun eine Menge Ideen in ein Programm umsetzen. Versucht es doch einfach mal selbst, zum Beispiel mit einem kleinen Rechenprogramm oder einem

Frage- und Antwortspiel. Es gibt ausreichend Programmierprobleme, die auf Lösungen warten. Viel Spaß dabei!

Unter die Arme gegriffen

Das Abtippen von Programmen, besonders von langen Basic-Programmen, kann nervtötend sein. Trotz sorgfältigster Eingabe funktionieren Programme oft nicht. Irgendwo haben sich doch Fehler eingeschlichen und die Fehlersuche nimmt viel Zeit und Nerven in Anspruch. Was kann ich dagegen tun? Wenn Ihr die Diskette dieses Buches zur Hand nehmt und mit

```
LOAD "$",8   <RETURN>
```

und dann

```
LIST         <RETURN>
```

das Inhaltsverzeichnis auf den Bildschirm ruft, entdeckt Ihr ein Programm namens »Checksummer«. Dieses Programm ist für die Eingabe von Listings eine große Hilfe, da es Eingabefehler weitgehend ausschließt. Der Umgang mit »Checksummer« ist ein Kinderspiel. Das Funktionsprinzip ist folgendes: Bevor ein neues Programm eingegeben wird, laden wir Checksummer in den C 64 und starten es mit RUN. Bei jeder eingegebenen Befehlszeile des neuen Programms erscheint oben auf dem Bildschirm in eckigen Klammern eine Zahl, eine sogenannte Prüfsumme. Diese Prüfsummen müssen mit den im Buch abgedruckten Prüfsummen neben den Befehlszeilen übereinstimmen. Richtige Eingabe erzeugt immer korrekte Prüfsummen, bei einer falschen Zahl muß die Befehlszeile kontrolliert werden. Probieren wir jetzt den Checksummer aus. Zuerst legen wir die Diskette in das eingeschaltete Laufwerk und schreiben

```
LOAD"CHECKSUMMER",8
```

Die Disketten-Station beginnt zu surren und meldet sich nach kurzer Zeit mit READY. RUN eingeben. Der C 64 meldet sich mit:

```
CHECKSUMMER 64

EINEN MOMENT, BITTE...
CHECKSUMMER AKTIVIERT.

AUSSCHALTEN : POKE 1,55 ODER <RUN/STOP+RESTORE>

ANSCHALTEN : POKE 1,53

READY
```

Die Eingabehilfe ist nun aktiviert und steht uns zu Diensten. Geben wir die erste Zeile unseres Programms ein

```
5 REM *** EIS-KAUF-PROGRAMM ***
```

Nach dem Drücken der RETURN-Taste erscheint oben links auf dem Bildschirm die Zahl <38>. Das ist die Prüfsumme der Programmzeile 5. Sie muß mit der im Buch angegebenen Zahl übereinstimmen – sie stimmt! Jetzt geben wir die zweite Zeile ein:

```
10 PRINT "<SHIFT+CLR/HOME>"
```

Hier noch einmal der Hinweis. Ihr gebt nicht SHIFT + CLR/HOME ein, sondern drückt nach den Anführungszeichen gleichzeitig die Tasten SHIFT und CLR/HOME. Auf dem Bildschirm erscheint ein reverses Herz. Die Prüfsumme ist <254> und damit richtig. Wenn sie nicht mit der im Buch angegebenen Zahl übereinstimmt, müßt Ihr die Fehler mit Hilfe des Cursors berichtigen und wieder RETURN drücken. Eine neue Prüfsumme erscheint. Mit diesem Verfahren wird das gesamte Programm eingegeben, fehlerlos! Wenn Ihr fertig seid, könnt Ihr den Checksummer durch die Eingabe von

```
POKE 1,55
```

ausschalten. Ist doch eine tolle Sache, oder? Nie wieder Probleme mit der Eingabe von langen Programmen. Gewußt wie! Im 64'er-Magazin werden alle Listings mit Checksummer-Prüfsummen veröffentlicht. Nun könnt Ihr alle Listings aus dem 64'er fehlerfrei abtippen.

Das haben wir gelernt

Strukturiertes Programmieren: Das geplante Umsetzen einer Idee in ein Basic-Programm. Wir schreiben erst in Worten Schritt für Schritt auf, wie unsere Idee aussieht, dann wird überlegt, was der C 64 dabei genau tun soll. Dazu zeichnen wir ein Struktogramm, eine symbolische Programm-Abfolge. Jetzt erst werden die Schritte in Basic übersetzt.

Algorithmus: Eine Folge von Programm-Schritten.

REM: Mit diesem Basic-Befehl können Kommentare oder Bemerkungen ins Listing eingeflochten werden. REM-Zeilen werden vom C 64 ignoriert und haben keinen Einfluß auf den Programm-Verlauf.

Checksummer: Ein spezielles Programm aus dem 64'er-Magazin. Anhand von Prüfsummen kann überprüft werden, ob eine Programmzeile richtig abgetippt wurde.

Kapitel 8

Der Bildschirm wird aktiv

Wir beißen uns durch verschachtelte Schleifen. Sie bringen Leben auf den Bildschirm: Ein Ball läuft von der einen Seite zur anderen. Das alles mit einem Programm von nur 13 Zeilen. Das neue Wunder-Programm findet Ihr in Listing 8.1. Der Name ist »FIXER BALL«. Es scheint magische Kräfte zu haben, denn der bisher eher ruhige Bildschirm entpuppt sich als äußert flexibles Gebilde. Wenden wir uns zunächst dem Eintippen zu (auf Diskette unter gleichem Namen zu finden).

```
10 REM FIXER BALL                                <234>
20 PRINT "{CLR}"                                 <008>
25 FOR P=1 TO 10:PRINT "{DOWN}":NEXT             <242>
30 FOR BA=1 TO 39                                <227>
40 PRINT " Q{LEFT}";:REM BALL IST SHIFT/Q        <014>
50 FOR Z=1 TO 5                                  <030>
60 NEXT Z                                        <024>
70 NEXT BA                                       <118>
75 REM BALL DUEST NACH LINKS                     <225>
80 FOR BA=39 TO 1 STEP -1                        <086>
90 PRINT"{SPACE,2LEFT}Q{LEFT}";                  <055>
100 FOR Z=1 TO 5                                 <080>
110 NEXT Z                                       <074>
120 NEXT BA                                      <168>
130 GOTO 20                                      <060>

© 64'er
```

Listing 8.1: FIXER BALL bringt Leben auf den Bildschirm.

In Zeile 20 drücken wir gleichzeitig die Tasten [SHIFT] und [CLR/HOME].

Die Begriffe in den geschweiften Klammern geben immer Tastenkombinationen an, die gemeinsam gedrückt werden müssen. Ihr dürft Euch nicht wundern, wenn auf dem Bildschirm andere Zeichen auftauchen. Es sind sogenannte Grafikzeichen. Sie sind

Symbole für bestimmte Funktionen. Durch das Drücken der beiden Tasten SHIFT- und CLR/HOME entsteht zum Beispiel in Zeile 20 ein reverses Herz. Es ist das Zeichen für »Bildschirm löschen«.

Genau das gleiche passiert in den Zeilen 40 und 90. Auf dem Bildschirm erscheinen merkwürdig schwarze und weiße Zeichen. In Zeile 40 müssen wir folgende Tasten drücken:

1. Eine Leertaste (einmal auf die große SPACE-Taste drücken).
2. [SHIFT] [Q] (dadurch entsteht der Ball).
3. [SHIFT] [CRSR ->]: Es gibt zwei CRSR-Tasten, wir müssen die rechte von beiden nehmen. Diese Taste verschiebt den Ball nach links, davon gleich mehr.

Nummer 90 enthält die gleichen Tastenfunktionen. Hinter dem PRINT-Befehl drücken wir:

1. Eine Leertaste.
2. [SHIFT] und [CRSR ->].
3. [SHIFT] und [CRSR ->] (beide Male nehmen wir wie oben die rechte Taste).
4. [SHIFT] und [Q] (Ball).
5. [SHIFT] und [CRSR ->].

Bei beiden Zeilen ist Punkt 1 besonders wichtig, sonst funktioniert unser Programm nicht richtig. Alle Probleme sind beseitigt. Los geht's. Wir starten das Programm mit »RUN« und [RETURN]. Alles klappt vorzüglich: Auf dem Bildschirm erscheint ein Ball, der zwischen beiden Bildschirmrändern hin- und herspringt.

Enträtselung

Wie funktioniert's? Im Prinzip ist es ganz einfach. Stellen wir uns eine Lichterkette vor, wie sie aus vielen Diskotheken und Schaufenstern her bekannt ist. In dieser Lichterkette sind eine Reihe von Glühbirnen hintereinander angeschlossen. Wenn die Birnen nacheinander kurz aufflackern, entsteht der Eindruck, das Licht bewege sich in der Kette. Natürlich tut es das nicht, aber unsere Augen sind so träge, daß wir eine fließende Bewegung wahrnehmen. Nichts anderes geschieht auf unserem Monitor. Die Lichterkette ist eine Zeile auf dem Bildschirm, die Glühbirnen sind die 40 Spalten, in die eine Zeile aufgeteilt ist. Wir lassen den Ball zunächst in der ersten Spalte erscheinen und gehen dann in die nächste. Dort läuft dasselbe Spiel ab, bis wir in Spalte 40 angelangt sind. Von hier aus geht es in umgekehrter Richtung weiter. Alles, was wir noch tun müssen ist, unser Gedankenmodell in ein vernünftiges Programm zu packen.

1. Wir wollen, daß der C 64 39mal einen Ball aufflackern läßt. Deshalb brauchen wir eine Schleife, die sich 39mal wiederholt. Diese Schleife beginnt in Zeile 30

```
30 FOR BA=1 TO 39
```

2. Der Ball soll aufflackern, also in der ersten Schleife erscheinen, um dann in der nächsten Schleife wieder zu verschwinden. Das macht Zeile 40:

```
40 PRINT "<SPACE><SHIFT/Q> <RECHTER CRSR>";
```

Sehen wir uns mal genau an, was zwischen den Anführungszeichen steht. Der C 64 druckt diese Zeichen der Reihe nach auf den Bildschirm. Als erstes also ein Leerzeichen. Dieses hat für uns eine löschende Funktion, das heißt, sollte an dieser Position irgendein anderes Zeichen stehen, verschwindet es nun, da der C 64 es mit einem Leerzeichen überschreibt. An der nächsten Position druckt er nun einen Ball. Das nächste Zeichen ist der Schlüssel zum Erfolg. Es ist ein Steuerzeichen und veranlaßt den C 64 nicht, etwas auf dem Monitor auszugeben, sondern läßt ihn einen Befehl ausführen. In diesem Fall lautet der Befehl: Versetze den Cursor um eine Stelle nach links, also eine Position zurück. Das ist derselbe Effekt, als würden wir im Direkt-Modus einmal auf die Cursor-Tasten drücken. Wenn Ihr das mal überprüft, stellt Ihr fest, daß der Cursor nun genau auf derselben Position steht wie unser Ball. Mehr ist noch nicht geschehen. Das Löschen des Balles vollzieht sich erst in der nächsten Schleife, die mit NEXT Z eingeleitet wird. Dort trifft der C 64 erneut auf unseren PRINT-Befehl. Der Ball wird jetzt vom Leerzeichen überschrieben, also gelöscht (Bild 8.1). Damit haben wir unseren dritten Willen:

3. Der Ball soll in der nächsten Position (Spalte) so lange aufflackern, bis Position 39 erreicht ist. Das geschieht durch Wiederholen der Schleife wie bei Punkt zwei.

Der C 64 produziert also einen flitzenden Spielball, indem er in einer bestimmten Zeile nacheinander Bälle ausdruckt und gleichzeitig den Ball davor löscht.

Bild 8.1: Schritt für Schritt wandert der Ball nach rechts. Dabei wird vor dem Ausdruck des neuen Balls der alte gelöscht.

Bäumchen, Bäumchen wechsel dich

Ihr habt es sicher schon gemerkt: Auf diese Weise wandert der Fußball nur nach rechts. Sobald der Spielball auf der rechten Seite des Bildschirms angekommen ist, tritt der zweite Programmteil ab Zeile 80 in Kraft. Da der Ball wieder 39mal aufflackern soll, muß eine Schleife her. Sie beginnt in Zeile 80. Die wesentliche Änderung zum ersten Teil befindet sich in Zeile 90. Sie befördert den Ball in die entgegengesetzte Richtung, bis er wieder auf den Bildschirmrand trifft. Bild 8.2 erklärt uns diese Zeile.

Bild 8.2: Der Rückweg des Balls ist etwas komplizierter, weil der Cursor bei jedem Buchstaben eine Position nach rechts rückt.

Wir befinden uns mit dem Cursor bei Position 0. Zuerst wird ein Leerzeichen gedrückt, der Cursor rückt zu Nummer 1. Es folgt der Befehl, den Cursor zwei Schritte nach links zu bewegen (Stellung −1). In diese Position wird der Fußball gerückt. Der Cursor befindet sich jetzt in Position 0. (Hier müssen wir uns nun eines klarmachen: Nachdem in −1 der Ball gedruckt wurde, rückt der Cursor automatisch eine Spalte weiter nach rechts!). Er wird durch den letzten Befehl wieder eine Position nach links gerückt, so daß er von hier beginnt. Die Sache geht von vorne los. Die Leertaste löscht den »alten« Fußball in Stellung −1, dann geht der Cursor zwei Schritte nach links (Position −2) und druckt dort den neuen Spielball aus. Mit dem letzten Befehl rückt der Cursor wieder zu Nummer −2, von der aus der nächste Vorgang beginnt. Der Fußball geht nach und nach seinen Weg zum linken Bildrand zurück.

Turbo-Computer

Der C 64 kickt durch ständige Wiederholung der Programmzeile 40 den Fußball nach rechts. Die Rückfahrkarte ist Zeile 90, die ähnlich funktioniert wie Nummer 40. Ein Problem stellt sich hier. Wenn der Computer diese beiden Zeilen wiederholt, ohne Pause zu machen, würden wir den Ball kaum erkennen können: er wäre viel zu schnell.

Der C 64 rast so schnell durch ein Programm, daß das menschliche Auge ihm nicht folgen kann. Zu diesem Zweck ist in den Zeilen 50 und 60 ein kleiner Trick eingebaut.

Nach Zeile 40 muß der Computer durch Nummer 50 und 60 von 1 bis 5 zählen und darf erst mit dem Programm fortfahren. Der Fußball bleibt einen Moment länger an seinem Platz, so daß wir ihn gut erkennen können. Wer will, kann das Programm beliebig ändern: Man muß den Wert für Z in Zeile 50 vergrößern oder verkleinern. Achtung, in Zeile 100 geschieht das gleiche wie in 50, nur gilt diese Zeile für den Rückweg des Balls. Spielt mal ein wenig mit den Werten in beiden Zeilen, Ihr werdet eine Menge Spaß haben. Der C 64 gehorcht auf Knopfdruck.

Computertraining

Habt Ihr es gemerkt? Das Programm ist klar wie Kloßbrühe, kein Zeichen ist unverständlich, alles unter Kontrolle. Der Computer macht, was wir wollen und hat keine Flausen mehr im Kopf. Wir sind die Chefs von diesem Plastikkasten. Das Programm eignet sich hervorragend als Trainingsobjekt für unser Können. Kleine Änderungen im Programm bringen die erstaunlichsten Ergebnisse.

Die Geschwindigkeit des Balles läßt sich durch Vergrößern oder Verkleinern der Werte in den Zeilen 50 und 100 beeinflussen. Der Ball kann zum Beispiel schnell nach rechts, aber langsam nach links wandern, je nach Eingabe. Viel Spaß haben wir mit den unscheinbaren Semikolons der Zeilen 40 und 90. Ein Semikolon besagt, daß der nächste Befehl ohne Zwischenraum direkt an den zuletzt ausgeführten Befehl gehängt werden soll. Zuerst rufe ich das Programm mit »LIST« und [RETURN] auf den Bildschirm. Mit dem Cursor lösche ich in Zeile 40 das Semikolon und teile dem Computer durch die RETURN-Taste die Veränderung mit. »RUN« startet das Programm: Der Bildschirm zeigt interessante Dinge. Der C 64 druckt untereinander 39 Bälle aus, dann läuft der Ball von rechts nach links über den Bildschirm und die Sache beginnt von vorne. Die Lösung dieses Spiels ist einfach. Der Computer rückt normalerweise nach jedem ausgeführten PRINT-Befehl an den Anfang der nächsten Zeile und führt dort den neuen Befehl aus. Das Semikolon hat dieses Verfahren bisher verhindert. Jetzt rückt der Cursor nach jedem PRINT-Befehl eine Zeile weiter und druckt 39mal einen Ball aus, bis der zweite unveränderte Teil des Programms in Aktion tritt. Prima was? Eine andere Spielerei findet sich in Zeile 25. Durch P wandert der Cursor 10 Zeilen nach unten, erst dort wird der Ball lebendig. Wer den Ball nicht so tief auf dem Bildschirm haben will, verändert einfach den Wert für P und kann das Spiel in jeder beliebigen Zeile beginnen lassen.

Neues auf dem Programmier-Markt

Leute, etwas fällt uns auf: Bisher haben wir den PRINT-Befehl nur zum Ausdrucken von Buchstaben, Worten oder Zahlen benutzt. Jetzt sehen wir ihn in einer unbekannten Funktion. Er kann auch Cursor-Bewegungen ausführen. Diese Bewegungen werden im Programm mittels grafischer Zeichen gespeichert und sichtbar gemacht. Sie können in den Anführungszeichen im PRINT-Befehl verwendet werden und der C 64 führt das Ganze ohne Probleme aus. Die Handhabung dieser neuen Programmiertechnik ist leicht. Nach dem Öffnen der Anführungszeichen im PRINT-Befehl drückt Ihr einfach auf die Tasten, die Ihr braucht. Zum Beispiel eine der Cursor-Tasten. Wenn Ihr einen Ball braucht, drückt Ihr die benötigte Tastenkombination [SHIFT] und [Q] , fertig! Und nun viel Spaß beim Experimentieren.

Das haben wir gelernt

Bewegung: Mit Grafikzeichen schafft man Bewegung auf dem Monitor, durch Löschen des Zeichens an der alten Position und gleichzeitiges Setzen an der neuen.

Cursor-Steuerung: Der Cursor läßt sich innerhalb eines Programms positionieren, indem wir mit den Cursor-Tasten innerhalb der Anführungszeichens des PRINT-Befehls definieren, wo er hin soll.

Kapitel 9

Zufällig gelesen

In Kapitel 6 haben wir uns mit den verschiedenen Eingabemöglichkeiten von Daten befaßt: INPUT und GET. Jetzt wenden wir uns der Turbo-Ausführung eines Basic-Befehls zu: READ und DATA öffnen neue Wege.

Wir haben die Möglichkeiten von INPUT und GET kennengelernt und miteinander verglichen, beide haben Vor- und Nachteile. In der einen Situation ist der Einsatz von INPUT sinnvoll, in einer anderen ist GET angebrachter. Beide haben einen großen Nachteil: Die Verarbeitung umfangreicher Datenmengen ist mit ihnen nicht möglich. Uns fehlt ein Befehl, der Daten schnell lesen und verarbeiten kann. Setzen wir uns folgende Aufgabe: Fünf Ausdrücke sollen in einem Programm verarbeitet werden. Unter »Verarbeiten« verstehe ich zum Beispiel ausdrucken. Auf den bisher bekannten Wegen wird dieser Plan zu einem ellenlangen Programm. Jede Information muß einzeln eingegeben werden:

```
10 A$="UMSTAENDLICH"
20 B$="LANGWEILIG"
30 C$="KOMPLIZIERT"
40 D$="VERWIRREND"
50 E$="ZEITVERSCHWENDUNG"
```

Das sind die fünf Begriffe, die wir ausdrucken lassen wollen. Bisher haben wir sie nur eingegeben, jetzt müssen wir den PRINT-Befehl unterbringen:

```
60 PRINT A$
70 PRINT B$
80 PRINT C$
90 PRINT D$
100 PRINT E$
```

Frischen wir schnell unser Wissen auf. Alle Variablen mit einem $-Zeichen heißen String-Variablen und enthalten Zeichenketten. In dem obigen Programm werden den

Variablennamen (A$,B$ ...) einzelne Begriffe zugewiesen, die in der zweiten Programmhälfte ausgedruckt werden. Die Funktion des Listings ist klar: Die fünf Begriffe werden auf den Bildschirm ausgedruckt, allerdings benötigen wir zehn Befehle dazu. Das muß einfacher gehen. Testen wir mit RUN [RETURN] das neue Programm: es ist lang, aber funktioniert. Wie kann das Listing vereinfacht werden?

Kurz und bündig

Ein Arbeitsgang wird fünfmal wiederholt. Bei Wörtern wie »Wiederholen«, »Wiederholung« und verwandten muß es bei uns wie aus der Pistole geschossen kommen: eine Schleife muß her. Das Problem ist nur, wie bekommen wir fünf verschiedene Ausdrücke in unsere Schleife? Der READ-Befehl wird uns eine große Hilfe sein. READ heißt soviel wie »lies« und hinter DATA verbergen sich die zu lesenden Informationen. Tippt folgendes Programm ein:

```
10 FOR I=1 TO 5
20 READ A$
30 DATA PRIMA,KLASSE,SUPERSCHNELL,PROBLEMLOS,EINFACH
40 PRINT A$
50 NEXT
```

Nach Eintippen von RUN [RETURN] erscheint:

```
PRIMA
KLASSE
SUPERSCHNELL
PROBLEMLOS
EINFACH

READY
```

Wir haben das Programm auf fünf Zeilen verkürzt! Fünf Zeilen für den Ausdruck von fünf Begriffen, das Verhältnis kann sich sehen lassen. Schauen wir uns die Befehle an, zuerst interessieren nur die Zeilen 20 und 30. Zeile 20 hieße übersetzt: »Lies A$«. Wir dürfen diesen Befehl ruhig so verstehen. Der C 64 sucht, nachdem er auf den READ-Befehl stößt, nach der ersten Programmzeile, die den Befehl DATA enthält. DATA heißt »Daten«. READ A$...DATA bedeutet demnach »Lies Daten und speichere sie unter der Variablen A$«. In unserem Beispiel braucht der C 64 nicht lange suchen. Er findet die gesuchten Daten bereits in der nächsten Befehlszeile, Zeile 30. Dies muß nicht so sein, wir könnten die Daten auch in eine Zeile 60 packen, etwa so:

```
10 FOR I=1 TO 5
20 READ A$
30 PRINT A$
```

```
40 NEXT
50 REM *** DATEN ***
60 DATA DAS,IST,EINE,ANDERE,VARIANTE
```

Der C 64 findet die DATA-Zeile immer, egal wo sie steht. Die Begriffe in unserer DATA-Zeile gehören zur String-Variablen A$. Hier stecken die fünf Ausdrücke, die oben so mühevoll eingegeben werden mußten. Obwohl es sich hier eindeutig um Zeichenketten handelt, müssen sie in der DATA-Zeile nicht in Anführungszeichen stehen.

Um die Kernzeilen READ und DATA herum gruppieren sich die anderen Befehle. In Befehl 10 wird eine Schleife eingeleitet, die zusammen mit Zeile 50 fünfmal durchflogen wird. Zeile 10 brauchen wir, damit der Computer nicht nur einmal einen Wert aus der DATA-Zeile 30 liest. Bei jedem der fünf Durchgänge wird ein neuer Wert gelesen, so erscheinen die Begriffe der Reihe nach auf dem Bildschirm. Der C 64 merkt sich, wo er in einer DATA-Zeile den letzten Wert geholt hat und geht beim nächsten Durchgang um eins weiter. Befehl 40 druckt den jeweiligen Wert von A$ aus. Wir müssen uns das so vorstellen: Wir fliegen in Zeile 10 los und erhalten dort die Information, daß diese Schleife fünfmal durchflogen wird. In 20 bekommen wir den Befehl, den ersten Begriff für A$ aus der DATA-Zeile zu lesen: »PRIMA«. Befehl 40 läßt uns den Inhalt von A$ ausdrucken und 50 schickt uns zu 10 zurück, das Spiel beginnt von vorne. Beim nächsten Durchgang lesen wir den zweiten Begriff »KLASSE«. Ist das nicht ein toller Trick?

Bewährungsprobe

Der READ-Befehl erspart uns eine Menge Arbeit. Sehen wir uns ein weiteres Beispiel an. Wir dürfen uns nicht verwirren lassen. In dem neuen Programm kommt ein Befehl vor, den wir uns später genauer vornehmen: der POKE-Befehl. Im Moment müssen wir nur soviel wissen: Unser C 64 besteht aus einer Menge einzelner Kästen, Speicher genannt. Jeder Kasten hat eine bestimmte Funktion und enthält eine Zahl. Mit Hilfe der Zahlen wird der C 64 gesteuert. Der POKE-Befehl ermöglicht es, die in den einzelnen Speichern enthaltenen Zahlen durch andere zu ersetzen. Die Speichernummer 53280 zum Beispiel ist der Speicher für die Farbe des Bildschirmrahmens (Bild 9.1).

	Rahmenfarbe	Hintergrundfarbe
53279	53280	53281

Bild 9.1: Im Speicherplatz 53280 liegt die Information für die Rahmenfarbe. POKE kann diesen Inhalt ändern.

Probiert mal kurz folgende Zeile aus:

```
POKE 53280,1  <RETURN>
```

Die Farbe des Bildschirmrahmens verändert sich! Wir haben mit POKE 53280 den Farb-Speicher geöffnet und eine neue Zahl, die Eins, hineingetan. Diese Ziffer ist der Wert für Weiß, der Rahmen wird weiß (keine Angst, alle Operationen lassen sich durch kurzes Aus- und Anschalten des Computers wieder rückgängig machen!). Wir können 16 Farben darstellen, die Ziffern 0 bis 15 stehen für die verschiedenen Farben. Die Sache mit den Speichern und dem POKE-Befehl sehen wir uns in einem späteren Kapitel an. Im Moment müssen wir uns nur eines merken: Im Speicher 53280 bewirken die Zahlen 0 bis 15 eine Veränderung der Bildschirmrahmenfarbe. Das nutzen wir jetzt aus.

```
NEW
10 FOR X=1 TO 15
20 READ A%
30 POKE 53280,A%
40 PRINT 53280 A%
50 FOR I=1 TO 500:NEXT
60 NEXT X
70 DATA 1,12,3,5,6,15,7,8,10,4,11,13,9,2,14
```

Bevor das Programm gestartet wird, sehen wir uns die Befehle genau an.

Zeile 10: Zähle für X von 1 bis 15. Beginne mit der Ziffer »1« und gehe dann in die nächste Zeile.

Zeile 20: Lies den ersten Wert für A% aus der DATA-Zeile. Zuerst also die 1. Lege den Wert 1 unter der Variablenbezeichnung A% in deinen Speicher und gehe weiter.

Zeile 30: Schreibe in den Speicher für die Rahmenfarbe den Wert von A% (im ersten Durchgang die 1, dann 12, 3 ...).

Zeile 40: Schreibe 53280 und dahinter den jeweiligen Wert von A% (beim ersten Durchlauf also »53280 1«. Ihr erinnert Euch, die Zahl hinter dem Komma der Speichernummer 53280 bewirkt eine Veränderung der Rahmenfarbe. Der Computer druckt in dieser Zeile die zur aktuellen Rahmenfarbe gehörige Zahl aus).

Zeile 50: Zähle für I von 1 bis 500. Die Zeile hat für den Programmablauf eine Stopp-Funktion. Der Computer zählt in etwa fünf Sekunden von 1 bis 500. Solange bleibt die neue Rahmenfarbe auf dem Bildschirm sichtbar. Ihr könnt Euch die Funktion dieser Zeile ganz einfach klarmachen, indem Ihr den Wert 500 ändert. Je größer diese Zahl ist, desto länger bleibt jede Rahmenfarbe sichtbar. Wenn Ihr Zeile 50 völlig aus dem Programm werft, verändern sich die Rahmenfarben mit einer irrsinnigen Geschwindigkeit. Probiert das einmal aus.

Zeile 60: Gehe zurück in Zeile 10, zähle das nächste X (jetzt X=2) und gehe in die nächste Zeile. In Zeile 20 liest der Computer den zweiten Wert für A% aus der DATA-Zeile: eine neue Farbe erscheint.

Dieses Spielchen läuft 15mal hintereinander ab. Jedesmal mit einer neuen Farbe. Einige werden jetzt sagen: »Ja, aber das geht auch ohne READ, nämlich so:

```
10 FOR X=1 TO 15
20 POKE 53280,X
30 PRINT 53280 X
40 FOR T=1 TO 500: NEXT T
50 NEXT X
```

Und dann sogar noch mit einer Zeile weniger.« Nun, das ist nicht ganz richtig, denn mit einer einfachen Schleife sind wir an eine bestimmte Folge von Farben gebunden, nämlich erst die Farbe für 1=Weiß, dann für 2 Rot usw.

Bei unserer Lösung mit DATA können wir die Farbfolge frei wählen, zum Beispiel erst die Farbe 1=Weiß, dann die für 12=Grau, ganz nach Belieben.

Der READ-Pfeil trifft immer

Die Arbeitsweise von READ ist faszinierend. In der DATA-Zeile werden die Informationen einfach hintereinander aufgelistet. Es werden keine Anführungszeichen verwendet.

Es reicht, wenn jeder Wert durch ein Komma vom nächsten getrennt ist. Nach dem letzten Begriff jedoch darf kein Komma stehen, sonst geht das Programm in die Hose.

Unser C 64 merkt sich genau, an welcher Position in der DATA-Zeile er sich gerade befindet. Das Ganze können wir mit einem Pfeil vergleichen, der mit jedem gelesenen Wert von A% eine Position weiterrückt (Bild 9.2). Beim ersten Wert steht der Pfeil auf der 1, beim zweiten auf der 2 und so weiter.

Auf diese Weise irrt sich der Computer nie, jeder Wert wird genau einmal ausgelesen, dann kommt der nächste.

Die Sache wäre verstanden, experimentieren wir ein wenig herum. So ein bißchen Training kann nie schaden! Aufgabe: Wandle das Programm mit den Bildschirmfarben in ein immer fortlaufendes Programm um.

Kein Problem, einfach eine neue Zeile anhängen:

Bild 9.2: Der READ-Pfeil rückt bei jedem gelesenen DATA-Wert eine Position weiter.

```
80 GOTO 10
```

Mit dieser Zeile müßte das Programm eigentlich endlos durchlaufen und immer wieder die 15 Farben ausdrucken. Ich tippe RUN ein.

Ein Schaudern überkommt mich, der Computer bricht nach der Zahl 15 das Programm unverschämterweise ab. Er meldet sich mit

```
?OUT OF DATA ERROR IN 20
READY
```

Die Fehlermeldung heißt soviel wie: In Zeile 20 sind keine lesbaren Daten mehr vorhanden. Bisher hat alles so prima geklappt, wieso kapiert der C 64 meine neue Befehlszeile nicht? In Zeile 20 befindet sich der READ-Befehl, der wiederum liest aus Befehl 70 die vorgeschriebenen Werte. Beim ersten Durchlauf bis 15 klappt es doch, wieso nicht beim zweiten? Da kommt mir die Erleuchtung: der READ-Pfeil! Die Fehlermeldung entsteht durch den READ-Pfeil. Er bleibt nach dem ersten Durchlauf auf der Zahl 14 (dem letzten Wert der DATA-Zeile) stehen und kehrt nicht automatisch an den Anfang der Zeile zurück! Beim zweiten Durchlauf kann kein Wert mehr gelesen werden, weil der Pfeil bereits auf der letzten Position steht. Unser Computer meldet: keine Daten vorhanden. Der Befehl, der den Pfeil auf den ersten Wert (die 1) zurückstellt, ist schnell gefunden: RESTORE. Wenn der Computer im Verlauf eines Programms auf den Befehl RESTORE trifft, so setzt er den READ-Zeiger auf den ersten

Wert der DATA-Zeile und das Programm ist wieder voll einsatzbereit. Machen wir die Probe aufs Exempel. Das vollständige Programm lautet:

```
10 FOR X=1 TO 15
20 READ A%
30 POKE 53280,A%
40 PRINT 53280,A%
50 FOR I=1 TO 500:NEXT I
60 NEXT X
70 DATA 1,12,3,5,6,15,7,8,10,4,11,13,9,2,14
80 RESTORE
90 GOTO 10
RUN
```

Die Sache funktioniert planmäßig. Das Programm läuft endlos weiter und läßt sich nur durch die RUN/STOP-Taste unterbrechen. Schon im nächsten Kapitel lernen wir eine weitere praktische Anwendung von READ und DATA kennen, und zwar in einem Musik-Programm.

Immer mehr Basic

Mit DATA können wir eine beliebige Folge von Werten im Programm verarbeiten. In unserem Fall haben wir ganz willkürlich die Rahmenfarbe des Bildschirms geändert. Nun stellt sich die Frage, ob wir auch den C 64 selbst entscheiden lassen können, in welcher Folge sich die Farbe des Rahmens ändern soll. Die Antwort gibt ein Befehl, der zum Computer schlecht paßt. Es handelt sich um den Befehl für den »Zufall«. Ja, Ihr habt richtig gelesen, mit diesem Befehl verwandelt sich der C 64 in einen Zufallsgenerator. RND(X) läßt den Computer aus einem Bereich »zufällig« Zahlen auswählen. Ein Beispiel ist der beste Lehrmeister.

```
10 D=RND(1)
20 PRINT D
30 GOTO 10
RUN
```

Dieser kleine Dreizeiler hat es aber in sich: Der Bildschirm füllt sich in Windeseile mit Zufallszahlen zwischen null und eins. Laßt Euch von der Schreibweise nicht verwirren. Der C 64 verschluckt die Null am Anfang und schreibt statt 0.907885285 ».907885285«. Gelegentlich tauchen in der Ziffernkolonne Zahlen auf, die mit »E« enden. Dahinter verbirgt sich ein Problem, das uns im Moment nicht interessiert. Wir beachten diese Zahlen einfach nicht. Viel wichtiger ist das Zustandekommen der einzelnen Werte. RND ist die Abkürzung von »RANDOM«, was auf deutsch »zufällig« heißt. RND(1) liefert uns Zufallszahlen zwischen null und eins. Die Frage ist: Wie erhalte ich größere

Werte und wie werde ich den Rattenschwanz nach dem Komma los? Am einfachsten wäre das Einsetzen eines höheren Wertes für 1 in Zeile 10. Leider klappt das nicht: Der C 64 druckt weiterhin nur Zahlen unter eins aus.

Es gibt eine zweite Möglichkeit: die Werte multiplizieren. Wir multiplizieren dazu die erhaltenen Zahlen zum Beispiel mit zehn, das Ergebnis sind Zufallszahlen zwischen null und neun. Achtung, das ist kein Schreibfehler. Wie wir eben gesehen haben, sind die erhaltenen Zufallszahlen immer kleiner als null.

Multiplizieren wir sie mit einer Zahl, so kann das Ergebnis nie größer oder gleich dieser Zahl sein. Multiplizieren wir eine Zufallszahl mit 10, erhalten wir Ergebnisse zwischen 0 und 9. Probieren wir es aus:

```
10 D=RND(1)
20 PRINT (D*10)
30 GOTO 10
```

Das Ergebnis ist nicht übel. Hier ein Probeausdruck meines Bildschirminhaltes:

```
2.77644135
.926713817
5.34902723
.14708266
6.52576545

BREAK IN 20
READY
```

Die Neuheit findet sich in Zeile 20. Befehl 10 liefert wie gewohnt Zahlen zwischen null und eins. In Zeile 20 werden diese Zufallswerte durch die Klammer mit zehn multipliziert. Wundert Euch nicht, wenn Ihr andere Zahlen auf dem Bildschirm habt. Das ist bei Zufallszahlen der Sinn der Sache.

Einige Unschönheiten müssen beseitigt werden: Die Ratttenschwänze nach den Kommas und die Zahlen unter eins müssen verschwinden. Unser Ziel ist es, Zufallszahlen zwischen eins und neun zu erzeugen. Der Befehl INT ist der Retter in der Not. Er schneidet die unbeliebten Nachkommazahlen ab (Bild 9.3): Aus 2.88988899 macht er 2, aus 9.111111112 entsteht 9. INT wird schlicht und ergreifend vor die Kommazahl gesetzt.

Bild 9.3: INT schneidet die Nachkommastellen eines Wertes ab.

Diese Tatsache prüfen wir an einem kleinen Beispiel nach. Schreibt

```
PRINT INT (2.88988899)  <RETURN>
```

Auf dem Bildschirm erscheint

```
 2

READY
```

INT ist ebenfalls eine Abkürzung und steht für »Integer«, was auf deutsch »ganze Zahl« bedeutet. Ihr erinnert Euch, in Kapitel 5 kam dieser Begriff schon vor. Integer-Variablen enthalten immer ganze Zahlen. Bauen wir den neuen Befehl sinnvoll in das Programm ein:

```
10 D=RND(1)
20 PRINT INT(D*10)
30 GOTO 10
RUN
```

Freude überkommt mich. Der Bildschirm zeigt nur ganze Zahlen zwischen null und neun. So, das war bisher recht trockener Tobak. Was können wir mit dem RND-Befehl alles anstellen?

Labyrinth mit Zufallsgarantie

Der Zufallsgenerator findet in Computerspielen Anwendung. Ein kleines Beispiel schauen wir uns in Form des nächsten Programms an. Es lautet:

```
NEW
10 PRINT CHR$(205.5+RND(1));
20 GOTO 10
RUN
```

Freunde, in diesem kurzen Programm sammelt sich eine ganze Menge unseres Wissens an. Jedes mit dem Computer darstellbare Zeichen hat eine eigene Nummer, einen sogenannten Code. Das wissen wir seit dem Kapitel über Variablen. Der CHR$-Befehl macht aus einer Nummer das dazugehörige Zeichen. In Zeile 10 verbinden wir dieses Wissen mit dem Zufallsgenerator. Auf dem Bildschirm entsteht ein Labyrinth, das sich nur durch Drücken der RUN/STOP-Taste beenden läßt. Die Funktionsweise dieses kleinen aber ergiebigen Programms haben wir schnell verstanden. Zum Verständnis helfen zwei Befehlszeilen.

```
PRINT CHR$(205) <RETURN>
PRINT CHR$(206) <RETURN>
```

Hinter den beiden Zahlen verbergen sich Grafik-Zeichen. Durch Zeile 10 addiert der C 64 im obigen Programm zu dem Wert 205.5 Zahlen zwischen null und eins. Die erhaltenen Werte bewegen sich von 205.5 bis 206.5. Für den Ausdruck nach CHR$ ignoriert der Computer die Nachkommastellen und druckt das zugehörige Zeichen aus: entweder 205 oder 206. Ein Labyrinth entsteht. Ist das nicht faszinierend? Zwei Programm-Zeilen verwandeln unseren Basic-Kameraden in ein Grafik-Wunder. Wir sehen, unser bisheriges Wissen wird immer wieder auf die Probe gestellt. Nichts darf vergessen oder zum alten Eisen gelegt werden, weil wir unsere Erkenntnisse immer wieder brauchen. Alles kein Problem für uns.

Zufallsfarben

Die Zufallszahlen können wir in unser Programm zur Änderung der Rahmenfarbe einsetzen. Das Programm sieht aus wie folgt:

```
10 POKE 53280,INT(RND(1)*16)
20 FOR I=1 TO 500
30 NEXT
40 GOTO 10
```

In Zeile 10 wird durch POKE 53280 die Rahmenfarbe geändert. Die Farbe wird durch INT(RND(1)*16) zufällig bestimmt. RND(1)*16 ermittelt eine Zufallszahl zwischen 0 und 15. Da wir aber nur »glatte« Zahlen brauchen können, schneidet INT alles ab, was hinter dem Komma steht. Der Rest dieses Programms ist uns bereits bekannt. Der »Normalverbraucher« kann mit dem RND-Befehl eine ganze Menge machen. Aber, das muß man ganz klar sagen, es gibt auch Grenzen. Wen es interessiert, der sollte weiterlesen. Alle anderen verpassen nicht viel, wenn sie jetzt zum nächsten Kapitel blättern.

Was Ihr nicht unbedingt wissen müßt

Was es mit dem Zufall wirklich auf sich hat, zeigt folgendes Listing:

```
10 PRINT RND(-1)
20 GOTO 10
```

Mit wunderschöner Regelmäßigkeit erhalten wir immer dieselbe »Zufallszahl«, nämlich »2.99 E-8«. In Zeile 10 fällt auf, daß hinter dem RND in Klammern eine Zahl mit einem Minuszeichen, also eine negative Zahl, steht. Genau hier liegt der Hase im Pfeffer. Beim RND(X)-Befehl sind die Zahlenwerte ziemlich bedeutungslos. RND(1) und RND(314) ergeben beide dieselbe Folge von »Zufallszahlen«, die direkt nach dem Einschalten des C 64 mit ».185564016« beginnt.

Der C 64 errechnet nämlich die Zufallszahlen nach einer bestimmten mathematischen Formel (Folge). Dabei geht er von einer Basiszahl aus. Sie ist sozusagen der »Keim«, aus dem Zahlen entstehen. Diese entstehen aber nicht zufällig, sondern eben in strenger Reihenfolge, gemäß der Formel. Ermitteln wir eine Zufallszahl mit:

```
10 Z=RND(1)
20 PRINT Z
30 GOTO 10
```

würden wir feststellen, daß sich die Reihenfolge der Zahlen nach 65536 Durchläufen wiederholt. Das liegt unter anderem am C 64 und auch an der Formel, beziehungsweise an der Basiszahl, aus der die Zufallszahlen ermittelt werden. Diese Basiszahl können wir jedoch mit einem negativen Wert hinter RND verändern. RND(-1) ändert dann die Basiszahl in »2.99 E-8«. Folgendes Programm verdeutlicht unser neues Wissen:

```
10 X=RND(-1)
20 PRINT X;
30 Y=1
40 PRINT RND(Y);
50 GOTO 10
```

Nachdem Ihr es mit RUN gestartet habt, stellt Ihr fest, daß die »Zufallszahl« X immer »2.99 E-8« ist. Klar, denn wir haben ja gerade festgestellt, daß RND(-1) die Basiszahl für Zufallszahlen festlegt. Von derselben Basiszahl ausgehend, sind eben auch die Folgen der Zufallszahlen bei jedem Programmdurchlauf gleich. Ändert einmal die Zeile 30 in:

```
30 Y=314
```

Nach dem Start mit RUN hat sich nichts an den Zufallszahlen geändert. Es sind immer noch die gleichen wie im ersten Programm. RND(1) und RND(314) ergeben dieselben Zufallszahlen. Ihr seht, meine Behauptung ist richtig. Heißt das, es ist Essig mit dem Zufall beim C 64?

Nein, RND(0) ist schon recht zufällig. Der C 64 hat in seinem Inneren eine Uhr, die nach seinem Einschalten anfängt zu laufen. RND(0) ermittelt die Zufallszahlen aus dem Stand der Uhr. In diesem Fall sind die Zufallszahlen wirklich zufällig. Probiert es aus mit:

```
10 X=RND(0)
20 PRINT X
30 GOTO 10
```

Nach jedem Neustart erscheint eine neue Folge von Zufallszahlen. Allerdings ist RND(0) nur mit Einschränkungen zu genießen: Es können nur 256 verschiedene Zufallszahlen ermittelt werden. Auch RND(0)*500 erzeugt 256 verschiedene Zufallszahlen.

Nun stecken wir in einer Sackgasse. Wir wissen, die Zahl hinter RND spielt keine Rolle, viel wichtiger sind die Vorzeichen. Weder RND(1) oder RND(-1) noch RND(0) verschaffen uns eine vernünftige Lösung. Ein kleiner Befehl kommt uns zu Hilfe. Mit TI kann die Zeit der internen Uhr des C 64 ausgelesen werden. Direkt nach dem Einschalten des C 64 steht diese Uhr auf null, beginnt dann aber sofort zu laufen. Leider läuft sie nicht in unserer gewohnten Weise, also in Sekunden, Minuten und Stunden. Sie zählt in sechzigstel Sekunden. Eine Minute zum Beispiel sind 3600 sechzigstel Sekunden. Doch das soll uns im Moment kalt lassen. Denn wir wollen nicht die Uhrzeit haben, sondern richtige Zufallszahlen.

Wichtig ist in diesem Zusammenhang, daß der C 64 eine interne Uhr hat, die ständig läuft und deren Uhrzeit in sechzigstel Sekunden angegeben wird. Mit TI können wir diese Uhrzeit auslesen und in unseren Programmen verwenden. Mit RND(-TI) erhalten wir eine recht willkürliche Basis für Zufallszahlen. Probiert es aus mit folgendem Programm:

```
10 PRINT RND(-TI);
20 GOTO 10
```

Ihr seht, wir erhalten bei jedem Programmdurchlauf einen anderen Startwert. Dieses Wissen läßt sich prima auf unsere Zufallszahlen anwenden:

```
10 Z=RND(-TI)
20 PRINT INT(RND(1)*10);
30 PRINT Z
40 GOTO 10
```

Auf diese Weise erhalten wir Zufallszahlen, die wirklich zufällig sind. Zwar werden sie von derselben Formel errechnet, jedoch bei jedem Programmdurchlauf mit einer anderen Basiszahl, also mit unterschiedlichen Keimen.

Das haben wir gelernt

READ: Liest Daten aus einer DATA-Zeile.

DATA: Enthält Daten, die das Programm für seine Arbeit braucht. Das können Zahlen, aber auch Zeichenketten sein. Die Daten werden mit dem Befehl READ ausgelesen.

RND(X): Erzeugt eine Zufallszahl im Bereich von 0 bis 1. Der Zahlenwert X ist dabei ohne Bedeutung. Wichtig ist er erst als negative Zahl. Sie setzt einen neuen »Keim«, mit dem neue Zufallszahlen errechnet werden.

Kapitel 10

Meister der Musik

Musik ist das Zauberwort. Mit ihr zeigt unser Computer-Freund seine künstlerische Ader. Ein Programm erzeugt mit verschiedenen Eingaben die tollsten Töne. Ob Glöckchen oder Preßlufthammer, mit dem C 64 ist alles möglich.

Wir wenden uns dem Thema »Sound-Programmierung« zu (»Sound« heißt auf deutsch Ton, Klang, Geräusch). Wie erzeugt der C 64 Töne, wie kann ein Computer Töne produzieren? Bevor diese Frage behandelt wird, spielen wir ein bißchen. Ich habe ein Programm für Euch, mit dem wir eine Menge Spaß haben werden (Listing 10.1). Das neue Programm findet Ihr unter dem Namen »TONSCHAU« auf der Diskette. Diesem Programm bringen wir gleich im wahrsten Sinne des Wortes die Flötentöne bei, und nicht nur die! Das wird spaßig.

Handarbeit

Hier einige Tips für die Arbeit mit Programmen, besonders für Musik-Programme. Falls der Computer beim Programmablauf stänkert, zum Beispiel mit Fehlermeldungen wie »ILLEGAL QUANTITY ERROR IN« oder dem leider altbekannten »SYNTAX ERROR IN«, macht Ihr folgendes:

1. Programm mit LIST auf den Bildschirm rufen.
2. Die nach »IN« angegebene Zeilennummer mit dem Cursor anfahren und die falschen Zeichen berichtigen. Bei »ILLEGAL QUANTITY ERROR« müßt Ihr besonders aufpassen. Diese Fehlermeldung deutet eine falsche Zahlenangabe in den DATA-Zeilen an. Auf deutsch bedeutet diese Meldung: »Es wurde ein zu großer Wert eingegeben«. Überprüft die DATA-Zeilen deswegen ganz genau, habt Ihr alle Kommas richtig gesetzt? Beim Vergessen eines Kommas entstehen sehr große Zahlen: Zum Beispiel 17103 statt 17,103. Der Computer kann mit diesen Werten nichts anfangen und fängt an zu mosern. Wenn alles nichts hilft, müßt Ihr das gesamte Programm von neuem durchsehen.

3. RETURN-Taste drücken!!!
4. Wenn das Programm erneut gestartet werden soll, drückt [SHIFT] [CLR/HOME]. Auf den gelöschten Bildschirm könnt Ihr jetzt RUN eingeben und das Programm mit [RETURN] starten.

```
100 REM---------------------------          <193>
110 REM TONSCHAU                             <159>
120 REM---------------------------          <213>
130 S=54272                                  <150>
140 READ A,D,SU,R,C,P,F,M,FF,FR,ML           <187>
150 POKE S+5 ,16*A +D                        <174>
160 POKE S+6 ,16*SU+R                        <037>
170 POKE S+2 ,P AND 255                      <085>
180 POKE S+3 ,P/256                          <133>
190 HI=INT(F/256):LO=F-256*HI                <142>
200 POKE S+0 ,LO                             <240>
210 POKE S+1 ,HI                             <018>
220 POKE S+22,FF                             <056>
230 POKE S+23,FR                             <154>
240 POKE S+24,ML                             <224>
250 GET A$:IF A$="" THEN 250                 <220>
260 : POKE S+4,C OR 1                        <191>
270 : FOR I=1 TO M:NEXT I                    <161>
280 : POKE S+4,C AND 254                     <193>
290 GOTO 250                                 <052>
300 REM---------------------------          <139>
310 REM PARAMETER                            <224>
320 DATA 0,10,0,10:REM A D S R               <165>
330 DATA 16        :REM CONTROL-BYTE         <222>
340 DATA 2048      :REM PULSWEITE            <129>
350 DATA 40000     :REM TONHOEHE             <148>
360 DATA 100       :REM VERZOEGERUNG         <216>
370 DATA 50        :REM FILTERFREQUENZ       <033>
380 DATA 0         :REM FILTERRESONANZ       <148>
390 DATA 15        :REM MODUS/LAUT           <209>
```


Listing 10.1: Das Programm TONSCHAU ist ein kleines Klangwunder.

Alles klar? Hinein ins Vergnügen.

Ton-Konzert

Nach dem Eintippen von RUN passiert zunächst nichts. Drückt auf irgend eine Taste: Ein Ton erklingt, bei jedem erneuten Drücken geschieht das gleiche. Probiert mal eine Taste mit Wiederholungsfunktion aus. Zum Beispiel die SPACE-Taste: wenn sie längere Zeit gedrückt wird, wiederholt sich der Ton automatisch.

Das Programm wird mit der Taste RUN/STOP unterbrochen. Da haben wir ein tolles Ding entdeckt. Ein kurzes Programm verwandelt den unscheinbaren C 64 in ein Klanginstrument. Das Beste kommt noch. Umstellen einiger Zahlen im Musik-Programm ermöglicht jede Menge verschiedener Töne und Klänge. Die Tabelle 10.1 zeigt einige Möglichkeiten.

Register	A S+5	D S+5	SX S+6	R S+6	Control-Byte C S+4	Puls-weite P S+2 S+3	Ton-höhe F S S+1	Ver-zögerung M	Filter-Fre-quenz FF S+22	Filter-Re-sonanz FR S+23	Modus Laut ML S+24
Glöckchen	0	10	0	10	16	x	40000	100	x	0	15
Oboe	8	7	10	8	64	250	7500	500	x	0	15
Fagott	8	7	10	8	64	250	2500	750	x	0	15
Zungenpfeife	8	0	15	10	48	x	400	1000	x	0	15
Banjo	0	8	0	8	32	x	7500	30	50	241	111
Feder	0	8	0	9	32	x	750	35	x	0	15
Preßlufthammer	0	0	15	10	80	2100	200	2000	x	0	15
Schuß	0	8	0	10	128	x	10000	50	x	0	15
Starkstrom	0	0	15	0	128	x	100	2000	x	0	15
Düsenflugzeug	0	0	15	13	128	x	3000	3000	50	241	31
Rakete	0	0	15	15	128	x	1000	3000	10	241	31

x = Ohne Wirkung (Parameter muß nicht eingestellt werden)

Tabelle 10.1: Der C 64 wird zum Synthesizer. Austauschen der Werte in Listing 10.1 genügt.

Wir können sowohl ein Glöckchen erklingen lassen, als auch eine Rakete starten. Die Sache funktioniert ganz einfach. An dieser Stelle zwei Tips: Das Programm wird mit RUN/STOP unterbrochen und kann mit LIST auf den Bildschirm gerufen werden. Falls Ihr nur die Zeilen bis 150 sehen wollt, gebt Ihr einfach LIST-150 ein. Bei Eingabe von LIST300- erscheinen alle Zeilen ab 300.

In Zeile 310 unseres Musik-Programms steht »REM PARAMETER«. Parameter sind Positionen, in die verschiedene Werte eingetragen werden können. Verändern der Zahlen in den Zeilen 320 bis 390 ruft die gewünschten Klänge hervor. Die Bezeichnungen der Parameter befinden sich hinter den Zahlen. »REM A D S R« zum Beispiel bedeutet, daß der erste Wert in Zeile 320 (hier also 0) der Parameter A ist, der zweite D (10) und so weiter. Zeile 330 enthält den Wert für das Control-Byte. Alle diese Bezeichnungen finden sich in Tabelle 10.1 wieder. Wenn die Werte der Tabelle in das Programm eingegeben werden, so ertönt der gewünschte Laut. Mit folgendem Beispiel lösen wir einen Schuß aus. Die einzutragenden Werte werden dazu der Reihe nach eingegeben. Achtung: Nach jeder neuen Befehlszeile muß RETURN gedrückt werden. Die Zeilen 320 bis 390 sehen jetzt so aus:

```
320 DATA 0, 8, 0,10: REM A D S R
330 DATA 128       : REM CONTROL-BYTE
340 DATA 2048      : REM PULSWEITE
350 DATA 10000     : REM TONHOEHE
360 DATA 50        : REM VERZOEGERUNG
370 DATA 50        : REM FILTERFREQUENZ
380 DATA 0         : REM FILTER-RESONANZ
390 DATA 15        : REM MODUS/LAUT
```

Überall da, wo in der Tabelle ein Kreuzchen steht, kann der alte Wert übernommen werden: Er beeinflußt das neue Geräusch nicht. Ganz besonders wichtig: Die Kommas in Zeile 320 dürfen nicht gelöscht werden, deswegen kontrolliert sie vor dem Start. RUN startet das Programm, [RUN/STOP] beendet es. Noch ein Tip, bevor Ihr richtig loslegt, wartet bei Starkstrom, Düsenflugzeug und Rakete, bis der Ton verklungen ist. Es kann sonst passieren, daß der Ton »stehenbleibt«, er hört nicht auf zu klingen. In diesem Fall habt Ihr das Programm mit [RUN/STOP] in der Befehlszeile unterbrochen, in der der Ton gehalten wird. Das ist nicht schlimm, verwirrt aber ein wenig. Viel Spaß beim Ausprobieren.

Zettelfarben

Der C 64 kann eine ganze Reihe von Klängen und Tönen produzieren, Tabelle 10.1 hat das bewiesen. Uns stellt sich die Frage, wie der C 64 die verschiedenen Töne herstellen kann. Mit diesem Problem nähern wir uns dem Thema »Datenverarbeitung des C 64«, das später genauer drankommt. Im Moment genügen uns folgende Informationen.

Ein Computer besteht aus vielen einzelnen »Speicherplätzen«. Sie haben die Funktion von Parkbuchten, die für ganz bestimmte Informationen reserviert sind. In die Speicherplätze legt der C 64 alle Eingaben ab, zum Beispiel Programme. Habt Ihr Euch schon einmal gefragt, wie sich der C 64 Dinge merken kann? Die Antwort ist: er legt sie in seinem Speicher ab, sozusagen dem Gehirn des Computers. Die einzelnen Bereiche des Speichers sind streng unterteilt, jeder hat eine bestimmte Funktion. Stellt Euch vor, der Speicher des C 64 ist eine riesige Tiefgarage mit einer Menge Etagen. Jede hat eine bestimmte Funktion. Die Etage, die wir gleich kennenlernen werden, ist die für Musik: ein für die Erzeugung von Tönen reservierter Bereich im Speicher.

Innerhalb dieses Bereiches befinden sich die Parkbuchten (Speicherplätze), die die Merkmale der Musik aufnehmen. Soll sie hoch oder tief sein oder soll sie wie eine quäkende Ente klingen, der man auf den Fuß tritt? Zusätzlich können wir festlegen, wie laut der Ton erklingt und wie viele Töne produziert werden sollen: Besteht das Lied aus 30 oder 500 Tönen? All das wird in den Speicherplätzen festgelegt.

Den Inhalt eines Speicherplatzes können wir uns als einen Zettel vorstellen, auf dem eine Zahl steht. Wenn der Inhalt des Speicherplatzes verändert werden soll, benötigen wir spezielle Befehle.

Der wichtigste dieser Befehle ist der POKE-Befehl. Er öffnet eine Parkbucht, nimmt den alten »Zettel« heraus und legt einen neuen hinein. Den POKE-Befehl lernen wir später ganz genau kennen. Hier nur eine kleine Einführung.

Eben haben wir festgestellt, daß es im Speicher des C 64 eine Menge einzelner Speicherplätze gibt. Speicherstelle Nummer 53280 zum Beispiel ist für die Rahmenfarbe des Bildschirms zuständig.

In ihr liegt normalerweise ein Zettel mit der Aufschrift »14«. Diese Zahl bedeutet: Die Rahmenfarbe ist Hellblau.

Zur Änderung des Rahmens können wir uns eine von 16 verschiedenen Farben aussuchen. Die Werte 0 bis 15 in Speicherstelle 53280 legen die gewünschte Farbe fest.

Wenn uns die »14« (Hellblau) nicht gefällt, können wir das sehr schnell ändern, Austauschen des Zettels genügt.

```
POKE 53280,7
```

bedeutet: Öffne Speicherplatz 53280 und lege die Zahl 7 hinein (die 7 bedeutet Gelb, vorher lag an dieser Stelle die Zahl 14 für Hellblau). Drückt jetzt die RETURN-Taste und staunt, wie schnell das geht: Die Rahmenfarbe wird Gelb. (Im Kapitel über PEEKs und POKEs findet Ihr eine Liste aller darstellbaren Farben und die dazugehörigen Zahlen.)

Musikalischer Baustein

All das brauchen wir jetzt für die Sound-Programmierung. Die Speicherplätze für die Musik-Programmierung befinden sich in einem elektronischen Baustein, er ist das Haus aller Musik-Schubladen: das Sound Interface Device (SID: Bild 10.1).

Eben haben wir von einer Tiefgarage gesprochen, in der es für bestimmte Funktionen festgelegte Parkplätze gibt. Der SID ist die »Parketage«, in der sich die Speicherplätze für die Sound-Programmierung befinden.

Bild 10.1: Der SID enthält die Speicherplätze der Sound-Programmierung.

Die erste Parkbucht im SID nennt sich Basisadresse (auf deutsch in etwa »Grundadresse«, die Adresse, von der aus wir mit dem Zählen beginnen). Sie hat die Nummer 54272. Von jetzt an werde ich diese Basisadresse S+0 nennen: S+0=54272. (Wundert Euch nicht über die null. Weshalb diese Schreibweise sinnvoll ist, werden wir gleich sehen.)

Mit dem C 64 können wir drei Stimmen gleichzeitig produzieren, sozusagen drei Instrumente auf einmal spielen lassen. Stellt Euch vor, Ihr singt mit zwei Freunden gleichzeitig. Jeder von Euch dreien singt etwas anderes. Egal, ob es schön klingt oder nicht, es ist dreistimmig.

Die Informationen über den Klang jeder einzelnen Stimme lagern in je sieben Parkbuchten (Register). Unter einem Register verstehen wir eine einzelne Parkbucht. Das Wort Register wird häufig im Zusammenhang mit der Sound-Programmierung verwendet. Der Commodore reserviert für jede Stimme sieben Parkbuchten (Register). Die Sache können wir uns so vorstellen: Vor uns steht eine Reihe von 21 Parkbuchten. Die erste hat die Nummer 54272 (die Basisadresse). In ihr und den sechs Parkbuchten danach liegen die Informationen der ersten Stimme, die nächsten sieben beinhalten die Daten der zweiten Stimme und in den letzten sieben Buchten liegen die der dritten.

Die erste Stimme, nur die interessiert uns im Moment, hat also die Register von S+0 bis S+6 und die Nummern 54272 bis 54278. In die Register kommen Zettel, die den gewünschten Ton genau festlegen. Bevor wir uns anschauen, was auf den Zetteln steht, machen wir einen Ausflug in die Welt der Musik. Nicht jeder ist ein kleiner Mozart.

Wellensalat

Jeder kennt das: Wenn irgendwo ein Klavier herumsteht, entsteht der unwiderstehliche Drang, darauf herumzuklimpern. Wer weiß jedoch, was im Inneren des Klaviers vorgeht? Es geschieht folgendes: Der Tastendruck schlägt einen kleinen Hammer auf eine Stahlsaite, die daraufhin in Schwingung gerät. Sie sendet einen Ton aus.

Stellt Euch ein Schwimmbecken vor, in dem seit Stunden keiner drin war. Das Wasser ist flach und ruhig. Steckt Ihr Euren Arm rein, wird das Wasser wellig. Ihr habt einen Impuls gegeben, der sich im Wasser ausbreitet. Wir sehen das in Form von Wellen. Der Klavierton ist nichts anderes als eine Welle, die sich durch die Luft bewegt. In der Fachsprache heißt dieser Begriff »Schallwelle«. Durch das Hin- und Herschwingen der Saite werden Impulse in die Luft gegeben, wie bei Eurem Arm im Wasser. Die Geschwindigkeit, mit der die Saite schwingt, nennt man »Frequenz«. Sie wird in Hertz angegeben. 440 Hertz (Abkürzung »Hz«) bedeutet, daß die Schallwelle beziehungsweise die Saite 440mal in einer Sekunde schwingt. Die Schallwelle gelangt nun in unser Ohr und wir nehmen sie als einen Ton wahr. Wir Menschen hören Schallwellen als Töne, wenn sie mit einer Geschwindigkeit im Bereich zwischen 20 und 20000 Hz schwingen. Eine Schallwelle mit einer hohen Hz-Zahl hören wir als hohen Ton, eine mit niedriger Hz-Zahl als tiefen Ton. Die im Klavier angeschlagene Saite schwingt in einer bestimmten Frequenz. Dicke Klaviersaiten langsamer (tiefer Ton), dünne Saiten schnell.

Fassen wir noch einmal zusammen. Die Klaviersaite fängt durch den Hammer an zu schwingen. Dieses Schwingen erzeugt die Tonhöhe, je schneller eine Saite schwingt, desto höher ist der Ton. Stellt Euch mal neben Vaters Stereoanlage und dreht laut auf. Hohe Töne tun in den Ohren weh und tiefe Töne könnt Ihr fast spüren: sie lassen das Bierglas auf dem Tisch vibrieren. Schaut in Tabelle 10.1. Das Glöckchen hat eine »Tonhöhe« von 40000. Es klingt hell und fein, während Starkstrom mit »Tonhöhe« 100 ein tiefes Surren erzeugt. Da klingelt es doch plötzlich im Köpfchen. Haben wir nicht eben gesagt, der Mensch kann nur Töne hören, die eine Frequenz zwischen 20 und 20000 Hertz haben? Wieso hat das Glöckchen dann eine Frequenz von 40000? Ein Druckfehler? Weit gefehlt, der Wert, der im Register steht, nennt sich Registerwert und unterscheidet sich von der wahren Frequenz. Zwischen den beiden besteht folgende Beziehung:

```
Frequenz=0.058717*Registerwert
Registerwert=17.0309*Frequenz
```

Das bedeutet, das Glöckchen hat einen Registerwert von 40000, aber eine tatsächliche Frequenz von 40000*0.058717Hz=2348.68 (Hz).

Immer tiefer hinein

Da brummt mir doch der Schädel. All diese Definitionen und neuen Worte! Wenn Ihr aber in der Schule bei Physik auf dieses Thema zu sprechen kommt, könnt Ihr sagen: »Das kenne ich schon!« Zurück zur Musik in unseren Ohren: Wie produziert der C 64 einen Ton? Im Grunde genommen ganz einfach, die Tonhöhe wird in die dafür reservierten Register gepackt und fertig. Leider nicht ganz. Die Zahl 40000 ist zu groß für ein Register. Die höchste Zahl, die ein Register aufnehmen kann ist 255. Ein Fuß mit der Schuhgröße 55 paßt auch nicht in einen Schuh Größe 38. Was können wir tun? Das Problem ist klar: Die Zahl 40000 kann in dieser Form nicht in den Computer eingegeben werden, da ein einzelnes Register nur Werte bis 255 aufnehmen kann.

Das Register kann nicht vergrößert werden, aber die Zahl 40000 läßt sich in veränderter Form darstellen. Sie wird in zwei Werte zerlegt, die kleiner als 255 sind. Stellt Euch vor, ihr müßtet die Zahl 6410 in zwei Parkbuchten stecken. In jede einzelne passen aber nur zwei Ziffern rein. Ich würde die Zahl 6410 in der Mitte trennen, also in die Teile »64« und »10«. Die erste »neue« Zahl käme in eine Parkbucht mit der Aufschrift »Hoch«, die zweite in eine mit der Aufschrift »Tief«. Sollte ich nun jemandem die ursprüngliche Zahl zeigen müssen, wäre es kein Problem. Ich würde zuerst die Ziffern aus der Parkbucht mit der Aufschrift »Hoch« herausholen und dann die Ziffern der zweiten mit der Aufschrift »Tief«. Mit meinen Aufklebern kann ich problemlos unterscheiden, welcher Zahlenteil zuerst steht. Die Ziffern in der Parkbucht mit der Aufschrift »Hoch« kommen vor denen der anderen. Die Arbeitsweise des C 64 ist etwas komplizierter, funktioniert aber nach einem ähnlichen Prinzip. Der Registerwert (zum Beispiel 40000 für das Glöckchen) wird nach bestimmten Formeln in zwei Zahlen zerlegt, die in die Register hineinpassen.

Das ist ein Hammer, was? Der C 64 ist nicht auf den Kopf gefallen. Für alles gibt es die richtige Lösung. Die beiden Zahlenteile heißen nun nicht mehr »Tief« und »Hoch«, sondern »Low-Byte« und »High-Byte« (von jetzt an nenne ich das Low-Byte »Lo-Byte« und das High-Byte »Hi-Byte«, so reden die echten Profis!). Damit haben wir die Inhalte der ersten beiden Register des SID. In die Basisadresse 54272 (=S+0) wird das Lo-Byte gelegt und Register S+1 (Speicherplatz 54273) nimmt das Hi-Byte auf. Das war der dickste Hund. Jetzt wird es wieder einfacher.

Ich fasse kurz zusammen: Da ein Register keine Zahl oberhalb von 255 aufnehmen kann, wird die Tonhöhe in zwei computergerechte Zahlen aufgeteilt und in die Register 54272 und 54273 gelegt (Vorsicht! Erinnern wir uns. Die Tonhöhe ist nicht gleich der tatsächlichen Frequenz des Tones. Wir haben vorhin gelernt, daß der Computer die eingegebene Tonhöhe (Glöckchen 40000) beim Programmablauf als reale Frequenz von 2348.68 Hz erklingen läßt). Diese beiden Speicherplätze bestimmen, wie hoch der produzierte Ton sein wird. Dieses war der erste Streich und der zweite folgt sogleich.

Ton ist nicht gleich Ton

Tonhöhe allein genügt nicht. Beim Klavier ist es genauso. Wer volle Suppe auf die Taste hämmert, bekommt einen anderen Klang als der, der leicht und zart tippt. Neben der Tonhöhe wird der Ton durch den Anschlag festgelegt. Soll der neue Klang hart und durchdringend oder weich und zart sein? Für diese Information sind im SID zwei Register festgelegt, die Speicherplätze 54277 und 54278 (S+5, S+6). Die in diesen Parkbuchten befindlichen Werte geben die Art des Tons an.

Leute, ein Ton ist eine komplizierte Sache. Welches Instrument hätten wir denn gerne? Mit dem Register S+4 können wir den Ton an den Klang verschiedener Instrumente angleichen. Der C 64 kann sowohl ein Klavier nachahmen als auch eine Flöte. Es kommt auf den Wert an, der sich in Speicherplatz 54276 (Register S+4) befindet.

Eine letzte Speicherstelle müssen wir uns noch ansehen. Sie hat die Nummer 54296 (also S+24) und ist der Lautstärkeregler unseres Computer-Orchesters. Er arbeitet wie der Lautstärkeknopf an einem Radio, man kann laut und leise stellen. Die Skala geht von 0 bis 15, wobei 15 »volle Lautstärke« bedeutet. Vorsicht, Register 24 bietet eine Neuheit: es ist für alle drei Stimmen zuständig! Die Lautstärke der drei Stimmen kann nicht einzeln eingestellt werden. Bevor wir uns in den vielen neuen Registern und Zahlen verirren, verschaffen wir uns mit Tabelle 10.2 einen Überblick. Sie listet die verschiedenen Register auf. Die bisher unbekannten Speicherplätze (S+2, S+3, S+21, S+22, S+23) brauchen wir momentan nicht.

Register im SID	**S = 54272**
	S+0 — Lo-Byte
	S+1 — Hi-Byte
	S+2
1. Stimme	S+3
	S+4 — Klangwahl
	S+5 — Art des
	S+6 — Anschlags
2. Stimme	S+7 bis S+13
3. Stimme	S+14 bis S+20
	S+24 — Lautstärke

Tabelle 10.2: Alle besprochenen Register auf einen Blick.

Hier der Überblick: Der C 64 kann drei Stimmen gleichzeitig produzieren. Jeder Stimme sind sieben Register zugewiesen. In diese sieben Register kommen die typischen Merkmale, wie zum Beispiel die Art des Klanges und die Art des Anschlags. Die

Tonhöhe wird in zwei Werte zerlegt und in die beiden ersten Register eingetragen. Unsere Beispiel-Stimme war die erste. Sie belegt die ersten sieben Register. Die Stimmen 2 und 3 verhalten sich genauso, nur haben die Register andere Nummern. Das Lo-Byte für Stimme 2 wird in Register S+7 untergebracht, das Hi-Byte in S+8 und so weiter. Neben den Registern der einzelnen Stimmen gibt es einige Schubladen für allgemeinere Informationen: Register 24 zum Beispiel ist für die Lautstärke zuständig.

Tonleiter klettern

Freunde, ich bin stolz auf Euch. Wir sind gerade dabei, eine sehr schwierige Sache zu verstehen. Unser Thema ist echter Profi-Stoff. Hier tauchen so viele neue Dinge auf einmal auf, daß wir es vielleicht mehrmals durchgehen müssen. Aber keine Sorge, wir schaffen alles.

```
10 REM EINE TONLEITER                                <040>
20 S=54272:REM BASISADRESSE SID                      <011>
30 POKE S+24,15:REM LAUTSTAERKE                      <156>
40 POKE S+5,9:REM ANSCHLAG                           <139>
50 READ X:READ Y:REM HI/LO-BYTE                      <004>
60 IF Y=-1 THEN END                                  <115>
70 POKE S+1,X:POKE S+0,Y:REM HI/LO-BYTE EI
   NPOKEN                                            <135>
80 POKE S+4,33:REM WELLENFORM, TON AN                <020>
90 FOR R=1 TO 100:NEXT:REM TONDAUER                  <003>
100 POKE S+4,32:REM TON AUS                          <192>
110 FOR R=1 TO 50:NEXT:REM PAUSE                     <114>
120 GOTO 50                                          <074>
130 DATA 17,103,19,137,21,237,23,59,26,20,
    29,69,32,219,34,207                              <101>
140 DATA -1,-1                                       <126>
```


Listing 10.2: Ein neues Listing vertieft unser Wissen.

Das neue Wissen nutzen wir jetzt in einem Programm: Listing 10.2. Es bietet einige Überraschungen: eine Tonleiter wird gespielt. Ihr findet es auf der Diskette unter dem Namen »EINE TONLEITER«. Mit RUN [RETURN] geht die Sache los: Der C 64 spielt 8 Töne hintereinander, die immer höher werden. Das macht mich neugierig. Was bewirken die verschiedenen Befehle?

Zeile 10: Die REM-Bemerkung erklärt uns die Funktion des Programms: es spielt eine Tonleiter. Eine Tonleiter ist eine Reihe von Tönen, die aufeinanderfolgen.

Zeile 20: Die Basisadresse wird festgelegt, S=54272. Das ist für uns sehr praktisch, denn jetzt müssen wir nicht immer 54272 eingeben, sondern nur noch S.

Zeile 30: Register S+24 ist für die Lautstärke zuständig. In diesem Befehl wird die Zahl 15 eingePOKEt: »Volle Lautstärke«.

Zeile 40: Diese Nummer legt den Anschlag des Tons fest. Soll er schnell und hart entstehen (ein harter Druck auf die Klaviertaste) oder klingt er weich und fein (leichter, sanfter Druck auf die Taste)?

Zeile 50: Diese Zeile holt sich die Hi- und Lo-Bytes aus Befehl 130. Dort sind alle acht Töne aufgelistet, immer zuerst ein Hi- (X) dann ein Lo-Byte (Y) und so weiter.

Zeile 60: Wenn der C 64 den Wert -1 aus der DATA-Zeile 140 liest, beendet er das Programm.

Zeile 70: Hi- und Lo-Byte werden in ihre Register eingePOKEt. Der C 64 hat in Zeile 50 das erste Werte-Paar gelesen: 17 für X und 103 für Y. Diese Zahlen setzt er jetzt ein und POKEt dadurch die Tonhöhe in ihre Register.

Zeile 80: S+4 ist für die Klangwahl zuständig. Der Wert 33 legt einen bestimmten Klang fest und »startet« den Ton. Wir können den Start mit dem Drücken der Klaviertaste vergleichen.

Zeile 90: Dieser Befehl läßt den C 64 von 1 bis 100 zählen. Während dieser Zeit können wir den in 80 gestarteten Ton hören. Ist 100 erreicht, springt der Computer in die nächste Zeile.

Zeile 100: Eben haben wir in S+4 den Ton gestartet, jetzt wird er wieder ausgeschaltet. Die Klaviertaste wird »losgelassen«.

Zeile 110: In dieser Zeile kann der Ton ausklingen. Der C 64 fährt eine Warteschleife und zählt bis 50. Wenn wir bei einem Klavier die Taste loslassen, klingt der Ton noch eine Weile nach. Genau dieser Effekt wird hier produziert: Der Ton kann ein wenig nachhallen.

Zeile 120: Der C 64 springt zur Zeile 50 zurück: Er liest ein neues Byte-Paar und beginnt so den Weg für den nächsten Ton.

Zeile 130: Hier sind alle Hi- und Lo-Bytes aufgelistet. Sie werden der Reihe nach gelesen und in Töne verwandelt.

Zeile 140: Die Werte, die den C 64 das Programm beenden lassen.

RUN ist der Startschuß. Zufriedenheit kreist über mir: Acht immer höher werdende Töne erklingen aus dem Lautsprecher meines Fernsehers. Der hätte sich auch nie träumen lassen, was ich alles mit ihm anstelle. Wer will, kann seine Fähigkeiten und Kenntnisse erproben. Bekanntermaßen ist Register S+24 der Lautstärkeregler. Probiert es mal mit einem Wert, der kleiner als 15 ist. Eine andere Spielerei bietet sich in den Zeilen 90 und 110 an. Das Programm läuft schneller ab, wenn in Befehl 90 statt 100

der Wert 50 steht. Eine Erhöhung verlangsamt die Tonleiter. Der gleiche Effekt zeigt sich bei der Nummer 110. Viel Spaß damit!

Freunde, das war schon echter Profi-Stoff. Eine harte Nuß, aber am Ende eine tolle Sache. Habt Ihr Geschmack daran gefunden und wollt mehr über die musikalischen Fähigkeiten wissen? Dann mache ich Euch folgenden Vorschlag: Wir lesen das Kapitel über PEEKs und POKEs und springen dann ins Musik-Kapitel des Fortgeschrittenenteils. Da schreiben wir unser eigenes Musik-Programm und erfahren viel mehr über die neue Programmiertechnik!

Das haben wir gelernt

SID: »Sound Interface Device« – Er ist der Baustein im C 64, der für alle Sounds verantwortlich zeichnet. Er kann einzelne Töne programmierbarer Klangfarbe und Tonhöhe erzeugen. Mittels des POKE-Befehls können einzelne Register gesetzt und somit einzelne Funktionen des SID aufgerufen werden. Die Musik-Programmierung ist etwas umständlich, da es keine gesonderten Befehle gibt. Selbst die Erzeugung des einfachsten Tons erfordert vier POKE-Befehle.

Kapitel 11

Computern ohne Basic – GEOS

Stellt Euch einmal vor, Ihr könntet den C 64 einschalten und ihn gleich bedienen. Ihr bräuchtet keine Basic-Befehle und doch wäre das Arbeiten am C 64 kein Problem. »Unmöglich« sagt Ihr? Falsch.

GEOS eröffnet faszinierende Möglichkeiten für den C 64. Gerade für uns Einsteiger ist GEOS eine tolle Sache!

Schon nach wenigen Minuten können wir Grafiken zeichnen, Texte schreiben oder mit zwei kleinen Bewegungen des Joysticks eine Diskette formatieren – ohne umständliche Befehle oder komplizierte Bedienungsanleitungen.

GEOS besteht aus einer unscheinbaren Diskette, die es aber in sich hat. Sie macht aus dem C 64 einen völlig neuen Computer. Schauen wir uns zunächst die neuen Möglichkeiten von GEOS an.

Seit kurzer Zeit gibt es eine deutsche GEOS-Version, die sich »GEOS 1.3« nennt. In dieser Version sind alle Befehle ins Deutsche übersetzt und es gibt eine deutsche Anleitung. Leute, ich kann Euch versprechen: gleich vergeht Euch Hören und Sehen.

Der einzige von »herkömmlichen Programmen« übriggebliebene Befehl ist:

```
LOAD "GEOS",8,1
```

Von jetzt ab brauchen wir keinen einzigen Basic-Befehl. Alles läuft über eine sogenannte »grafische Benutzeroberfläche«. Auf dem Bildschirm erscheint nach einer Weile die erste Seite des GEOS-Buches, die wir uns genau anschauen werden.

Bild 11.1: Eine grafische Benutzeroberfläche.

Ein völlig neues »Gesicht«

Auf Bild 11.1 sehen wir eine grafische Benutzeroberfläche. »Grafisch« deshalb, weil alle Programme und Funktionen zeichnerisch (grafisch) in Form von kleinen Symbolen auf dem Bildschirm dargestellt sind. Wir brauchen nun nicht mehr

```
LOAD"PROGRAMMNAME",8
```

einzugeben, um ein Programm zu laden. Mit einem Pfeil können wir die Symbole »anfahren« (markieren) und schon geht's los. Ich habe mich für Euch ein bißchen umgesehen: Auf großen, mehrere Tausend Mark teuren Rechnern ist diese Programmart weit verbreitet. Der C 64 muß sich hier keinesfalls verstecken.

Bevor wir gleich noch voll in die Möglichkeiten von GEOS einsteigen, lernen wir den »Verbindungsmann« zwischen uns und GEOS kennen, den schon erwähnten Eingabepfeil. Er wird von einem Joystick oder einer »Maus« gesteuert. Einen Joystick kennt jeder: Mit ihm steuert Ihr zum Beispiel die kleine Giana durch ihr abenteuerliches Leben. Der Joystick wird in den »Joystick-Port« auf der rechten Seite des C 64 eingesteckt. Die Maus funktioniert ähnlich wie der Joystick. Sie wird ebenfalls in den Joystick-Eingang gesteckt. Die Steuerung des Eingabe-Pfeils erfolgt durch das Bewegen der Maus auf der Tischplatte. Bild 11.2 zeigt Euch eine solche Maus. Sie hat auf ihrer Unterseite eine Kugel, die diese Bewegung auf den Bildschirm überträgt. Keine Sorge,

Ihr könnt für GEOS genausogut einen Joystick verwenden. Die Maus ist für die Steuerung des Eingabe-Pfeils etwas genauer, weil sie die Bewegungen exakter überträgt.

Bild 11.2: Eine Maus ist etwas exakter zu bedienen als der Joystick.

Jede kleine Bewegung des Joysticks oder der Maus verändert die Position des Eingabe-Pfeils auf dem Bildschirm. Alle Befehle werden dem Computer durch Drücken des Feuerknopfes mitgeteilt. Ein kleines Beispiel: Links unten auf dem Bildschirm sehen wir ein »Eselsohr«. Durch diesen Knick des »GEOS-Blattes« können wir eine weitere Seite des dargestellten GEOS-Buches erkennen. Wenn wir diesen kleinen Ausschnitt mit dem Eingabe-Pfeil anfahren und dann auf den Feuerknopf drücken, zeigt uns GEOS die dazugehörige Seite (der Vorgang des Anfahren-und-Feuerknopf-Drückens wird treffenderweise »Anklicken« genannt). Auf der neuen Seite befinden sich die Anwendungsprogramme GEOPAINT, GEOWRITE und DISKETTENKOPIER. Mit GEOPAINT werden wir nachher noch die tollsten Dinge malen. Es ist nämlich ein Zeichenprogramm.

Wenden wir uns einer ersten Demonstration von GEOS zu. Ihr alle kennt das nervende Problem des Disketten-Kopierens. Nehmt mal folgenden Fall an: Ein Freund von Euch besitzt ein tolles Spiel auf Diskette und Ihr möchtet Euch davon eine Kopie machen. Uns Einsteigern stellt sich die Frage, wie mache ich das? Mit GEOS kein Problem. Wir fahren mit dem Joystick-Pfeil das Zeichen von DISKETTENKOPIER an (es ist das erste von rechts!). Ein solches Programmsymbol nennen wir ab jetzt »Piktogramm«. Freunde, mir brummt der Schädel. Die neuen Begriffe geraten etwas durcheinander. Damit wir die neuen Ausdrücke nicht verwechseln und einen guten Überblick gewinnen, habe ich alle Begriffe in Tabelle 11.1 alphabetisch aufgelistet und erklärt. Wenn Ihr etwas nachlesen wollt, schaut in die Tabelle am Ende des Kapitels.

Zurück zum Kopier-Piktogramm. Nachdem wir den Pfeil auf das Piktogramm gefahren haben, drücken wir zweimal kurz hintereinander den Feuerknopf. Dieser »Doppelklick« lädt das Kopierprogramm. Der Bildschirm zeigt nach kurzer Zeit die Worte:

```
DISKETTENKOPIERPROGRAMM

ZU KOPIERENDE DISK IN LAUFWERK EINLEGEN
UND "RETURN" DRUECKEN, UM FORTZUFAHREN
```

Ist das nicht fantastisch? Von jetzt ab müssen nur die auf dem Bildschirm ausgedruckten Befehle ausgeführt werden und wir erhalten eine Kopie der tollen Spiele-Diskette. Die Sache ist ein Kinderspiel: Kein Ringen mit ellenlangen Basic-Befehlen, einfach ein Doppelklick und die Post geht ab!

Alles ganz easy!

So einfach ist das Kopieren einer Diskette mit GEOS. Ein weiteres Beispiel. Oben auf dem Bildschirm befindet sich die »Kommandoleiste«. Sie enthält das Befehlsmenü. Ihr erinnert Euch an Kapitel 6, in dem der Begriff »menügesteuert« auftauchte. GEOS ist das Musterbeispiel eines menügesteuerten Programms. In der Kommandoleiste sind eine Reihe von Funktionen untergebracht:

```
GEOS/DATEI/ANZEIGE/DISKETTE/SPEZIELL
```

Hinter jedem Menüpunkt verbergen sich »Untermenüs«. Die Untermenüs steuern die einzelnen Funktionen. Bild 11.3 zeigt das nach einmaligem Anklicken von DISKETTE erscheinende Untermenü. Es wird in der GEOS-Fachsprache »Pull-down-Menü« genannt.

Pull-down bedeutet soviel wie »Herunterziehen«, das Untermenü wird wie ein Rollo von oben nach unten gezogen. Mit den aufgeführten Befehlen kann zum Beispiel auf einfachste Weise eine Diskette formatiert werden. Ihr kennt den Befehl für die Disketten-Formatierung :

```
OPEN 15,8,15"N:name,id"
```

Der Befehl ist bekanntermaßen nicht gerade kurz und »anwenderfreundlich«. Mit GEOS sieht die Sache nun ganz anders aus. In dem nach DISKETTE erscheinenden Untermenü FORMATIEREN genügt Anklicken. Eine »Dialogbox« legt sich über den Bildschirm. Sie gibt uns Anweisungen für den Formatiervorgang. In ihr steht:

```
LEERDISKETTE IN LAUFWERK A
UND DISKETTENNAMEN EINGEBEN:
```

Mit Laufwerk A ist unsere Disketten-Station gemeint. Der Name der neuen Diskette wird über die Tastatur eingegeben und [RETURN] gedrückt, schon wird die Disketten-Station lebendig und formatiert die neue Diskette. Der umständliche Formatier-Befehl entfällt völlig. Eine Dialogbox ist ein Feld, in dem der Computer beziehungsweise GEOS von uns eine Information benötigt, die nicht mit dem Eingabe-Pfeil gesteuert werden kann: zum Beispiel der Name der neuen Diskette. Dieser muß über die Tastatur eingegeben werden. Bei Bedienungsfehlern verwendet GEOS ebenfalls Dialogboxen, in denen der gemachte Fehler aufgeführt ist. Wir sehen, GEOS sichert sich gegen jede Gefahr ab.

Neues von GEOS

Bisher haben wir uns in die Vorteile eingearbeitet, die GEOS sozusagen nebenbei liefert. Der Umgang mit den Disketten vereinfacht sich erheblich, weil wir nicht ständig irgendwelche Befehle auswendig lernen müssen. Der Eingabe-Pfeil und die Dialogboxen führen uns ohne großen Aufwand durch alle Möglichkeiten hindurch.

Bevor wir die auf der GEOS-Diskette gespeicherten Programme voll nutzen können, müssen wir uns mit dem Begriff »Backup« bekanntmachen. Unter einem »Backup« versteht man eine Sicherheitskopie, die alle Programme einer Diskette auf einer anderen sichert. Bei Datenverlust, zum Beispiel durch defekte Disketten, erspart das eine Menge Zeit und Ärger. Stellt Euch vor, Ihr arbeitet an einem Programm. Irgendwann macht Ihr eine Pause. Durch einen dummen Zufall schüttet Ihr Limonade über Eure Diskette. Und dann? Alle Arbeit war umsonst, denn dieser Datenträger ist hin. Eine zweite Diskette mit einem Backup, also einer Sicherheitskopie, hätte einiges leichter gemacht.

Zwei Dinge müssen wir beim Umgang mit GEOS beachten: Erstens, die Originaldiskette ist randvoll. Neue Texte oder Grafiken können auf ihr nicht gespeichert werden. Zweitens ist die Ursprungsdiskette besonders wertvoll: Nur mit ihr kann GEOS gestartet werden. Deshalb wird sie möglichst wenig Belastungen ausgesetzt.

Arbeits-Disketten für jedes Programm müssen her. Die Sache ist so, wenn wir mit GEOPAINT Bilder malen, wollen wir sie hinterher auch auf Diskette speichern. Die Originaldiskette bietet für solche Grafiken keinen Speicherplatz. Die Herstellung einer Arbeitsdiskette ist kein Problem, das läuft über die Joystick-Steuerung. Zuerst kopieren wir GEOPAINT auf eine Arbeitsdiskette. Dazu wird aber eine formatierte Diskette benötigt. Auf dem Bildschirm muß die Seite mit dem GEOPAINT-Piktogramm sichtbar sein. Das Piktogramm wird mit dem Joystick angefahren und einmal der Feuerknopf gedrückt. Das Piktogramm ist plötzlich schwarz. Nach etwa einer Sekunde (oder länger) drücken wir erneut den Feuerknopf: ein »Geister-Piktogramm« erscheint. Dieses Piktogramm können wir frei über den Bildschirm bewegen. Es wird unten am Bildschirmrand, unterhalb des GEOS-Buches »abgelegt«: Den Feuerknopf drücken, das

Piktogramm wird wieder normal sichtbar und der Joystick-Pfeil freigegeben. Jetzt müssen wir in der Kommando-Leiste DISKETTE und dann SCHLIESSEN anklicken. Der C 64 schließt die Originaldiskette und bereitet sich auf die neue Diskette vor. Wir nehmen die Urprungsdiskette aus dem Laufwerk und legen die leere, formatierte Diskette ein. Auf der rechten Seite des Bildschirms befindet sich ein Disketten-Symbol mit einem Fragezeichen. Anklicken dieses Symbols »meldet« dem C 64 die neue Diskette.

Auf dem Bildschirm erscheint die neue, noch leere Seite unserer zukünftigen Arbeitsdiskette. Wir führen einen weiteren Pausen-Doppelklick auf das am Bildschirmrand liegende GEOPAINT-Piktogramm aus: Anklicken, eine Sekunde warten, wieder den Feuerknopf drücken. Das beweglich gewordene Geister-Piktogramm ziehen wir links oben auf das leere Blatt und feuern einmal. Über den Bildschirm legt sich eine Dialogbox, die das Einlegen der Originaldiskette (»System« genannt) fordert. Wir tun das und klicken das »OK«-Feld an.

GEOS kopiert Stück für Stück GEOPAINT von der Originaldiskette auf die Arbeitsdiskette. Die beiden Disketten werden den Befehlen der Dialog-Boxen entsprechend gewechselt. Nach einer Weile begrüßt uns die Arbeits-Diskette mit dem GEOPAINT-Zeichen. GEOPAINT ist startklar.

Auf Leonardo da Vincis Spuren

Das Grafik-Programm GEOPAINT gibt uns die Möglichkeit, mit dem Computer farbige Bilder oder Grafiken zu zeichnen. Wir benötigen dazu weder Stift noch sonst ein Zeichengerät, der Joystick und ein geschicktes Händchen genügen. Der Weg in die Zeichenwelt führt über einen Doppelklick: das Piktogramm anfahren und zweimal direkt hintereinander den Feuerknopf drücken. (Ein kleiner Tip: Der Doppelklick erfordert ein wenig Übung. Manchmal reagiert der Feuerknopf nicht schnell genug und es entsteht ein Geister-Piktogramm oder das Piktogramm wird nur schwarz, sonst nichts.)

Ein Geister-Piktogramm legt Ihr einfach auf das Original-Piktogramm und drückt einmal den Feuerknopf. Bei einem verfärbten Piktogramm fahrt Ihr vom Symbol herunter und feuert einmal. Jetzt könnt Ihr erneut den Doppelklick probieren. Diese Probleme sind bei mir am Anfang öfter aufgetaucht, mit der Zeit verschwinden sie. Übung macht den Meister!

Die Disketten-Station rattert und surrt los. Auf unserem Bildschirm erscheint die Begrüßungsgrafik von GEOPAINT. Die Dialogbox fordert von uns eine Entscheidung: Wollen wir ein neues Dokument erstellen, ein existierendes öffnen oder nach Desktop verlassen.

Bild 11.3: Alle Funktionen lassen sich bei GEOPAINT von der Menüleiste (links) aus aufrufen.

»Desktop« ist die Seite, die nach dem Laden von GEOS erscheint. »Nach Desktop verlassen« hieße aus GEOPAINT aussteigen und wieder zum Anfangsbildschirm zurückkehren. Diese Arbeitstechnik der Dialogbox am Anfang eines Programms taucht bei GEOS häufig auf. Wir können in Ruhe entscheiden, ob wir eine alte Grafik vervollständigen, oder eine neue beginnen wollen.

Wir können momentan nur ein neues Dokument (sprich Grafik) eröffnen, da wir das Programm zum ersten Mal benutzen. Nach dem Anklicken von »Erstellen« erscheint eine weitere Dialogbox, die die Eingabe des Namens fordert. Wir nennen die neue Grafik »Test«. Jetzt geht es richtig los. Die Dialogbox verschwindet und macht dem »Zeichenblatt« Platz.

Der Joystick-Pfeil hat sich in einen kleinen Zeichenpfeil verwandelt. Auf der linken Seite in Bild 11.3 sehen wir eine Menüleiste. Hier können wir jede beliebige Zeichenart aussuchen. Anklicken des gewünschten Symbols eröffnet uns die tollsten Möglichkeiten. Im »Normalzustand« malt der Zeiger einen dünnen Strich auf den Bildschirm. Geschrieben oder gemalt wird durch Drücken des Feuerknopfes.

Einmaliges Drücken läßt den Pfeil schreiben, ein zweites Feuern schaltet wieder ab. Sobald der Joystick-Pfeil das Zeichenblatt verläßt, erhält er wieder seine bekannte Größe und alle Symbole können angeklickt werden.

Das zweite Symbol in der rechten Spalte zum Beispiel verwandelt den Zeiger in eine Sprühdose. Wir klicken es an.

Wenn jetzt der Feuerknopf gedrückt wird, erscheinen kleine »gesprühte« Felder auf dem Bildschirm. Die Kommandoleiste bietet eine Reihe von Befehlen:

GEOS/DATEI/EDITIEREN/OPTIONEN/SCHRIFTARTEN

Über die verschiedenen Menüpunkte öffnen sich Pull-down-Menüs, die keine Wünsche offenlassen. Ihr probiert die Möglichkeiten am besten selber aus. Bild 11.4 zeigt einen kleinen Überblick.

Bild 11.4: GEOPAINT ist ein vollwertiges Grafik-Programm. Hier einige Möglichkeiten.

GEOS-Schatztruhe

Auf der GEOS-Diskette befinden sich neben GEOPAINT eine ganze Reihe anderer Programme. Zum Beispiel das Textverarbeitungsprogramm GEOWRITE. Ein Textverarbeitungsprogramm bietet die Möglichkeit, den Computer wie eine hochmoderne Schreibmaschine zu verwenden. Texte können am Bildschirm geschrieben und wieder verändert werden, sie werden »bearbeitet« und ausgedruckt (wenn ein Drucker angeschlossen ist).

Der Vorteil liegt ja klar auf der Hand: Bei einer Schreibmaschine wird jeder Buchstabe sofort auf das Papier gedruckt. Textverarbeitungsprogramme drucken den Text erst aus, wenn er vollständig und für uns Benutzer zufriedenstellend ist. Die Handhabung von

GEOWRITE funktioniert nach dem Prinzip von GEOPAINT: Alle wichtigen Informationen und Befehle werden dem C 64 mit dem Joystick eingegeben.

GEOS bis zum Horizont

Die bisher besprochenen Programme befinden sich alle auf einer Diskette. GEOS ist eine ganze Serie.

Es gibt mittlerweile die verschiedensten Zusatzprogramme, alle funktionieren nach dem bekannten Prinzip: Eine nützliche Sache ist »GEODEX«. Auf der GEODEX-Diskette befinden sich mehrere Programme. Das Programm namens »GEODEX« ist eine Dateiverwaltung: es verwaltet Adressen inklusive Telefon und einer kleinen Überraschung. Alle Eintragungen können durch Gruppenzugehörigkeiten ergänzt werden. Nehmen wir an, wir wollen allen Mitgliedern des Fußballclubs 1. FC einen Rundbrief schreiben. Ein Rundbrief ist zum Beispiel die Einladung zur nächsten Vereinsparty. Der gleiche Brief geht an alle Mitglieder, nur die Adresse wird verändert. GEODEX sorgt ohne langes Suchen für die Ausgabe der richtigen Adressen.

Das ebenfalls auf der Diskette enthaltene Programm »GEOMERGE« bietet die Möglichkeit, in Zusammenarbeit mit GEODEX sämtliche Rundbriefe auf einen Schlag zu schreiben. Wer einmal eine solche Arbeit von Hand gemacht hat, lernt diese Ersparnis schnell schätzen. Nach verrichteter Arbeit kann sich der fleißige Vereinsleiter mit dem Kartenspiel »BLACKJACK« vom Streß erholen und gegen die Computerbank um hohe Einsätze spielen. Auf der Diskette befindet sich ebenfalls ein Kalender, der alle wichtigen Daten und Termine aufnimmt. Der Spieler kann ruhig loslegen, ein Blick in das Programm »KALENDER« läßt ihn keinen Termin vergessen.

Für alle, die viel mit Texten zu tun haben, bietet sich die Zusatzdiskette »WRITER'S WORKSHOP« an. Der Titel heißt übersetzt »Arbeitsplatz des Schriftstellers«. Auf der Diskette befindet sich ein sehr ausgefeiltes Textverarbeitungsprogramm. Es ist die Erweiterung des unter GEOWRITE bekannten Programms.

Texte können hervorragend bearbeitet werden, selbst das griechische Alphabet ist verwendbar. Einzelne Textabschnitte werden unterstrichen oder durch Fettdruck hervorgehoben, die dazugehörigen Befehle sind schnell gelernt. Ein besonderes Bonbon bietet die sogenannte »Suche-Ersetze-Funktion«.

Nehmen wir folgenden katastrophalen Fall an: Wir haben einen Text von zehn Schreibmaschinenseiten geschrieben. Der Text geht um das Thema »Nilpferde« (das sind die dicken Viecher aus dem Zoo, die fast immer nur im Wasser herumstehen. Sie werden auch »Flußpferde« genannt). Im Text kommt dieser Begriff ungefähr 30mal vor. Plötzlich stellen wir nun fest: Wir haben jedesmal »Nielpferd« statt »Nilpferd« geschrieben. In den Augen des Autors glänzt der Wahnsinn: Den langen Text auf den Begriff »Nielpferd« durchforsten und jedesmal den gleichen Fehler ersetzen! Nicht so mit

WRITER'S WORKSHOP. Die Suche-Ersetze-Funktion sucht selbständig »Nielpferd« und setzt »Nilpferd« ein. Ruckzuck ist das Problem gelöst.

Eine weitere Diskette aus der GEOS-Reihe räumt mit jeglichen Dateiverwaltungen auf. In Dateiverwaltungen werden alle möglichen Dinge verarbeitet: Adressen, Bücher, Briefmarken, Schallplatten, Videokassetten und so weiter. Das Problem ist, jeder Inhalt braucht ein eigenes Programm. Mit dem Dateiprogramm für meine Schallplatten kann ich keine Adressen verwalten und umgekehrt!

Es kann auch folgendes Problem geben: Ich will alle Adressen plus Geburtstage verwalten. Das Programm bietet nur Platz für die Eintragung von Adressen und Telefonnummern! »GEOFILE« schafft hier Abhilfe. Wir können unsere Dateiverwaltung selbsttätig zusammenstellen: So viele Eintragungsmöglichkeiten (»Felder« genannt) wie ich brauche. Nie wieder Probleme mit zu dürftigen Dateiverwaltungen! Eine praktische Sache.

Die letzte Zusatzdiskette, die ich hier auch noch vorstellen möchte, greift in das harte »Wirtschaftsleben« ein. In der weiten Welt da draußen gibt es einen Begriff, der sich »Tabellenkalkulation« nennt. Nehmen wir an, wir sind Besitzer eines Mofas oder Autos. Jeden Monat fallen feste Kosten wie Versicherung und Steuer an. Hinzu kommen sich verändernde Kosten: Benzingeld (hängt von der Menge der gefahrenen Kilometer ab), Kosten für Reparaturen und zum Beispiel der Preis für neue Reifen. In einer Tabellenkalkulation werden diese Kosten gegeneinander aufgerechnet.

Die monatlichen Benzinkosten durch die gefahrenen Kilometer geteilt, geben einen Überblick über die Kosten pro Kilometer. Wenn zusätzlich das Geld für Versicherung, Steuer und Reparaturen mit in die Kosten pro Kilometer einbezogen werden, erhalten wir die Gesamtkosten pro Kilometer. Jeden Monat ergeben sich neue Werte, das heißt: Bei jeder Veränderung eines Wertes muß die ganze Tabelle nachgerechnet und angeglichen werden. »GEOCALC« macht diesen Arbeitsgang überflüssig.

Wir erstellen ein einziges Mal die gültige Tabelle, GEOCALC rechnet alle Werte von alleine nach. Egal, wie oft sich der Wert für die gefahrenen Kilometer oder der Benzinpreis ändert. Den neuen Wert eingeben, fertig. Eine riesige Arbeitserleichterung.

Das haben wir gelernt

Anklicken: Der Eingabe-Pfeil wird auf einem grafischen Symbol positioniert und der Feuerknopf gedrückt. Das angeklickte Symbol wird für eine weitergehende Arbeit ausgewählt.

Backup: Eine Sicherheitskopie, die ein »faules« Leben an einem sicheren Ort führt. Sie wird nur verwendet, wenn die Originaldiskette kaputt ist oder wichtige Daten verlorengingen.

Desktop: Das Desktop ist der »Aktenordner« von GEOS. Auf dem Bildschirm wird es als ein »Buch« mit verschiedenen Seiten dargestellt. Es erscheint nach dem Starten von GEOS.

Dialogbox: Eine Dialogbox erscheint immer dann, wenn der Computer eine Information über die Tastatur benötigt, oder ein Bedienungsfehler vorliegt. Der Name einer neuen Grafik muß zum Beispiel über die Tastatur eingegeben werden.

Disketten-Symbol: Es erscheint rechts oben auf dem Bildschirm und zeigt den Namen der momentan bearbeiteten Diskette an. Wenn im Disketten-Symbol ein Fragezeichen steht, muß die neue Diskette durch Anklicken angemeldet werden.

Dokument: Jede neue Grafik oder jeder neue Text ist ein Dokument.

Doppelklick: Ein Doppelklick besteht aus zweimaligem Drücken des Feuerknopfes ohne Pause. Er startet die einzelnen GEOS-Programme über das Anklicken der Piktogramme.

Eingabepfeil: Er kann mit Hilfe des Joysticks (oder der Maus) über den Bildschirm bewegt werden und ist der »Kontaktmann« zwischen Anwender und GEOS. Alle Befehle werden durch ihn übermittelt.

Felder: Im Rahmen von Dateiverwaltungen wird mit Feldern gearbeitet. In die Eingabefelder werden die Informationen wie Wohnort, Straße, Telefonnummer eingegeben.

Geister-Piktogramm: Entsteht durch einen Pausen-Doppelklick auf ein Piktogramm und ist ein mit dem Joystick bewegbares, »durchsichtiges« Abbild des betreffenden Piktogramms.

Kommandoleiste: Ganz oben auf dem Bildschirm erscheinende Menüleiste, in der verschiedene Befehlsmöglichkeiten aufgelistet sind. Anklicken eines Kommandos führt in ein Untermenü (Pull-down-Menü).

Maus: Steuerung des Eingabepfeils, arbeitet genauer als der Joystick und ist für Grafik-Programme besonders geeignet. Eine kleine Kugel auf der Unterseite der Maus überträgt die Bewegungen auf den Bildschirm.

Menü: Ein Menü listet verschiedene Befehlsmöglichkeiten auf. Bei GEOS werden alle Menüs durch den Joystick gesteuert.

Menüpunkt: Ein einzelner Befehl aus einem Befehlsmenü.

Pausen-Doppelklick: Ein zweimaliges Anklicken eines Piktogramms, zwischen den beiden Klicks wird mindestens eine Sekunde gewartet.

Piktogramm: Programme werden bei GEOS in Form von Piktogrammen dargestellt. Das grafische Symbol, das die Funktion des Programms in Umrissen wiedergibt, nennt man Piktogramm.

Pull-down-Menü: Siehe »Untermenü«.

Tabellenkalkulation: Eine im Wirtschaftsleben häufig verwendete tabellarische Auflistung von Unkosten. Sie ermöglicht einen Überblick über steigende oder sinkende Ausgaben und Kosten.

Untermenü: Ein Untermenü kann auch Pull-down-Menü genannt werden. Es entsteht, wenn im Hauptmenü der Kommandoleiste ein Befehl gewählt wird. Der eingegebene Befehl wird genauer dargestellt.

Kapitel 12

Drucker: Vom Bildschirm aufs Papier

Leute, unser Computer fühlt sich einsam. Ihm sind die Hände gebunden, er kann seine tollen Fähigkeiten nur auf dem Bildschirm zeigen. Damit ist jetzt Schluß: Ein Drucker muß her und zwar hurtig.

Das ist leichter gesagt als getan. Kleine Informationsversuche meinerseits wurden von Händlern und Fachliteratur mit Begriffen wie Plotter, Schriftbild, Grafikfähigkeit, serielle Schnittstelle, Interface, ESC/P-Standard und ähnlichem Kauderwelsch untergraben. Nachdem ich mir das alles angehört hatte, mußte ich erst einmal ein Eis essen: Wie kann man sich nur so kompliziert ausdrücken. Das hat hiermit ein Ende! Hart aber herzlich begeben wir uns in den Drucker-Dschungel und stecken unsere zähe Nase in sämtliche Probleme. Dann mal ran an die Bouletten!

Holz vorm Kopf

Der erste Problem-Baum, den wir fällen werden, trägt die Aufschrift »Grafikfähigkeit« und »Schriftbild«. Bevor ich mir einen Drucker anschaffe, muß ich eins genau wissen: Was will ich mit dem Ding hinterher anstellen? Brauche ich den Drucker zum Briefeschreiben, will ich hauptsächlich Computer-Programme oder Grafiken ausdrucken? Zu jedem Vorhaben gibt es den richtigen Drucker-Typ und da wiederum Unmengen verschiedener Fabrikate. Früher war die Sache ganz einfach, da gab es Vaters mechanische Schreibmaschine, auf der man so schön herumhämmern konnte. Die machte tierischen Lärm und entsprechend viel Spaß, allerdings bekam man von den Dingern nach fünf Minuten verbogene Finger.

Beim Drucker gibt es eine Menge zu bedenken. Die »Grafikfähigkeit« ist für die weitere Drucker-Planung von großer Wichtigkeit. Unter Grafikfähigkeit versteht man die Fähigkeit des Computers zur Darstellung von Grafiken beziehungsweise Zeichnungen. Schauen wir uns den ersten Drucker-Vertreter an, der nicht grafikfähig ist, dafür aber andere Vorzüge hat: den Typenrad-Drucker.

Ein solcher Drucker arbeitet mit einem kleinen Rad, dem Typenrad, von dem dünne, eng beieinanderliegende »Speichen« abgehen. Am Ende jeder Speiche befindet sich ein Buchstabe. Die Buchstaben werden durch einen hinter dem Typenrad angebrachten Hammer einzeln gegen das Papier geschlagen. Wir müssen uns das so vorstellen: Wir drücken ein »A«. Das Typenrad dreht sich so lange, bis das »A« ganz oben ist, der Hammer schlägt gegen die Speiche und der Buchstabe wird gedruckt. Wenn wir jetzt ein »B« eingeben, dreht das Typenrad das »B« nach oben und der Hammer schlägt es auf das Papier. Das Rad dreht sich mit einer irren Geschwindigkeit. Auf ihm befindet sich ein bestimmter Zeichensatz, der nur durch Austausch des gesamten Rads verändert werden kann.

Ein solcher Drucker bietet das Schriftbild einer Schreibmaschine, im Grunde ist er eine an den Computer angeschlossene Schreibmaschine. Der große Nachteil: Er ist genauso wenig grafikfähig wie eine Schreibmaschine, mit Buchstaben können keine Zeichnungen ausgedruckt werden. Hinzu kommt ein im Vergleich mit anderen Druckern sehr schleppender Ausdruck.

Bild 12.1: Ein Matrix-Drucker.

Wenn Ihr die Karikatur Eures großen Bruders ausdrucken wollt, müßt Ihr einen anderen Drucker kaufen. Zum Beispiel einen Nadel-Matrix-Drucker (Bild 12.1). Was ist das schon wieder für ein Begriff? Ein Matrix-Drucker druckt einen Buchstaben nicht wie ein Typenrad auf einen Schlag aus, er »piekst« den Buchstaben in einzelnen Punkten auf das Papier. Das »Pieksen« übernimmt eine unterschiedliche Menge von Nadeln.

Die Arbeitsweise könnt Ihr Euch schnell klarmachen: Nehmt einen feinen Filzstift und »punktet« ein »A«, setzt eine Reihe von Punkten zu einem »A« zusammen. Genauso funktioniert der Matrix-Drucker. Ein 9-Nadel-Drucker zum Beispiel hat neun »Filzstifte«, die vom Drucker einzeln gesteuert werden. Damit sind wir beim Nachteil: Man kann hinterher ohne Probleme erkennen, hier hat ein Computer gearbeitet und keine Schreibmaschine (Bild 12.2). Solch ein Ausdruck hat keine »Briefqualität«.

```
Auf der anderen Seite besitzen Matrix-
Drucker teilweise eine sehr gute
Grafikfähikeit, die einzelnen Nadeln
erlauben den Ausdruck von Zeichnungen und
Grafiken. Hiermit könnt Ihr die
herrlichsten Karikaturen produzieren. Bei
moderneren Matrix-Druckern ist das Schrift-
Problem gelöst.
```

Bild 12.2: Der Ausdruck eines Matrix-Druckers: Deutlich sind die Punkte der Buchstaben zu erkennen.

Auf der anderen Seite besitzen Matrix-Drucker teilweise eine sehr gute Grafikfähigkeit, die einzelnen Nadeln erlauben den Ausdruck von Zeichnungen und Grafiken. Hiermit könnt Ihr die herrlichsten Karikaturen produzieren. Bei moderneren Matrix-Druckern ist das Schrift-Problem gelöst.

Bäumchen, Bäumchen falle um!

Die Lösung heißt »NLQ-Schrift«. NLQ bedeutet »Near Letter Quality« und heißt auf deutsch »fast Brief-Qualität«. Ein Matrix-Drucker mit NLQ-Funktion liefert ein Schriftbild, das die Drucker-Herkunft vergessen läßt. Die gute Schrift kommt folgendermaßen zustande: Der Drucker »piekst« den Buchstaben zunächst normal auf das Papier. Anschließend verschiebt er das Papier ein klein wenig nach oben und druckt den Buchstaben ein zweites Mal. Auf diese Weise schließt er die Löcher zwischen den einzelnen Punkten. Logisch gefolgert dauert dieser Ausdruck etwas länger, liefert allerdings eine hervorragende Qualität. Bild 12.3 zeigt den Vergleich der verschiedenen Schrift-Typen. NLQ ist einer der Begriffe, mit denen mich der Händler ständig beworfen hat. So einfach kann die Sache sein. Wandern wir weiter.

Star LC24-10
LQ-Courier
Courier kursiv
LQ-Prestige
LQ-ORATOR-SCHRIFT
LQ-Script-Schrift
Script kursiv
Outline
Shadow
Outline+Shadow
EDV-Schrift
EDV-Kursiv
Elite-Schrift
Schmalschrift
Breit
Fettdruck
Doppeldruck
Hoch- und tief
Überstrichen
Doppelt hoch

ABCDEFGHIJKLMNO
STUVWXYZÖÄÜöäüß
34567890+*?<>)(
$§"!^-.,;:_abcd
hijklmnopqrstuv
LQ 1:1 ▲

ABCDEFGHIJKLMNO
STUVWXYZÖÄÜöäüß
34567890+*?<>)(
$§"!^-.,;:_abcd
hijklmnopqrstuv
EDV 1:1 ▲

Aa
LQ 1:5 ▲

Bild 12.3: Schrift-Typen im Vergleich.

Der letzte Schrei sind 24-Nadel-Drucker (Bild 12.4). Sie arbeiten auf der gleichen Basis wie die 9-Nadel-Drucker, sind aber wegen der vielen Nadeln exakter als die »kleinen« Drucker. Beide Drucker-Typen gehören zu den Matrix-Druckern, weil die Nadeln in einer bestimmten Anordnung (Matrix) gruppiert sind.

Matrix-Drucker sind weit verbreitet, vor allem die 9-Nadel-Typen. In absehbarer Zeit werden sie von den 24-Nadel-Druckern eingeholt werden, die im Moment noch relativ teuer sind.

Matrix-Drucker können mehrere hundert Zeichen pro Sekunde ausdrucken. Ein Text von 2000 Buchstaben liegt in sechs bis sieben Sekunden vor.

Bild 12.4: Die neueste Errungenschaft: Ein 24-Nadel-Drucker.

Beenden wir unsere kleine Drucker-Schau mit der Vorstellung von »Plottern« und »Laser-Druckern«. Ein Plotter (Bild 12.5) ist ein Gerät, mit dem Grafiken von hoher Qualität »gezeichnet« werden können.

Bild 12.5: Ein Plotter ist ein hervorragender Grafik-Drucker.

Das Zeichnen übernimmt ein Stift, der in einer computergesteuerten Vorrichtung hängt. So ein Ding ist etwas für Profis, für uns ist es weniger interessant, denn es gibt bisher keinen Plotter, der vernünftig mit dem C 64 zusammenarbeitet. Ähnlich verhält es sich mit dem »Laser-Drucker«: ein sehr großer, teurer Drucker, der tolle Ausdrucke liefert. Für uns private Anwender ist er unerschwinglich, mehrere tausend Mark müssen ausgegeben werden.

Freunde, ich gebe zu, das Thema war bisher etwas abgehoben. Wir müssen durch den Dschungel: Da sind Begriffe wie »Plotter« und »Matrix-Drucker« lebensnotwendig! Machen wir weiter? Dann los.

Computer und Drucker, ein schwieriges Pärchen

Drucker kaufen, einstecken, alles paletti?! Die Sache ist leider etwas schwieriger. Drucker allein genügt nicht. Wenn ich zu einem Händler gehe und sage: »Henning braucht einen Matrix-Drucker, welcher ist gut?«, dann geht es mir schlecht. Der Händler erzählt mir etwas von Commodore-Druckern, Epson oder sonstigen Fabrikaten. Hinzu kommt die Frage: »Welches Interface hätten Sie denn gern?«, und ich gehe geknickt nach Hause, von wegen Interface. Auch der zweite Versuch ging in die Hose. Jetzt packt mich der Ehrgeiz: Welcher Drucker wird wie und wo an den C 64 angeschlossen und was brauche ich für den Anschluß? Kommt Zeit, kommt Drucker.

Der Anschluß eines Druckers an den Computer erweist sich als komplizierte Sache: Der Drucker ist fast ein eigener Computer, der programmiert und gesteuert werden kann. Es gibt eine wichtige Voraussetzung für den Zusammenschluß zweier Computer: Beide müssen die gleiche Sprache sprechen. Wenn mir ein Chinese zubrüllt, ich solle das Wort für »Reis« schreiben, verstehe ich kein Wort. Ich zeige ihm einen Vogel und gebe ihm den Stift, was er dann schreibt, kann wiederum ich nicht entziffern. Ähnliche Späße haben schon so manchem Drucker-Besitzer den letzten Nerv geraubt. Der Drucker schreibt den ganzen Text in eine Zeile oder druckt völlig unverständliche Symbole aus. Das Problem ist: Der Computer ist nicht richtig angepaßt.

Zunächst müssen wir uns um die Frage kümmern: Wie bekommt der Drucker Daten? Es gibt mehrere Möglichkeiten. Im C 64 gibt es einen geschlossenen Kreis von Daten. Die Informationen fließen zum Beispiel von der Tastatur zum Basic-Interpreter und zum Bildschirm. In diesen Daten-Kreis muß »eingebrochen« werden, damit der Drucker die nötigen Informationen erhält. Für den »Einbruch« gibt es drei Wege: User-Port, Erweiterungs-Port oder den sogenannten »seriellen« Ausgang, der im Moment von unserer Disketten-Station besetzt ist (siehe Kapitel 3). Wenn wir die Tastatur vor uns haben, befindet sich ganz links auf der Rückseite der User-Port, ganz rechts der Erweiterungs-Port und ungefähr in der Mitte der serielle Ausgang, an dem die Disketten-Station hängt. Am einfachsten anzuschließen sind die Drucker von Commodore. Die verbreitetsten sind die Modelle MPS 801 und MPS 803. Sie werden

einfach in den seriellen Eingang gesteckt und sind sofort funktionsfähig. Das klingt prima, hat aber leider einige Nachteile. Die Commodore-Drucker arbeiten mit der sogenannten »seriellen Datenübertragung«. Bei dieser Übertragungsart werden die Informationen Buchstabe für Buchstabe (sozusagen »in Serie«; eben seriell) an den Drucker übermittelt. Der Nachteil ist, die Übertragungsart ist sehr langsam und der Drucker schreibt entsprechend lahm: Zeichen für Zeichen tröpfelt aus dem Computer in Richtung Drucker. Der Vorteil: Die Anpassung entfällt, die Drucker schreiben problemlos Listings oder Texte.

Auf die Dauer überwiegen allerdings die Nachteile: Beide Drucker sind für die Textverarbeitung ungeeignet, weil das Schriftbild zu schlecht ist. Grafiken sind mit Einschränkungen möglich. Was passiert, wenn wir unseren Computer-Park um einen neuen Computer erweitern?

Die Commodore-Drucker können wir vergessen, sie laufen nur an Commodore-Computern.

Lange Leitung

Wie sieht es mit den anderen Druckern aus? Tja, hier wird die Sache knifflig. Einige werden an den Erweiterungs-Port angeschlossen, andere an den User-Port. Eins haben sie gemeinsam: Das Anschließen ist bedeutend umständlicher. Das hat seine Gründe:

Diese Drucker arbeiten mit »paralleler Datenübertragung« und besitzen eine sogenannte Centronics-Schnittstelle. (Die Schnittstelle ist der Platz, an dem Informationen aus dem Datenkreislauf herausgeholt oder hineingegeben werden können. Bei unserem Drucker ist sie die »Eingangstür« der zu druckenden Daten.)

Das heißt, diese Drucker verstehen nur parallel übertragene Informationen. Diese Übertragungsart beruht auf einem gleichzeitigen (parallelen) Datentransport von acht Informationseinheiten (Bits genannt; um die Bedeutung dieses Begriffs kümmern wir uns in Kapitel 14) über acht Kanäle zugleich.

Wie wir bereits wissen, druckt der Matrix-Drucker Zeichen aus, indem er Punkte in entsprechender Anordnung auf dem Papier ausgibt. Vergrößerten wir einen solchen Buchstaben, sähe er aus wie es Bild 12.6 zeigt. In einem 8 * 8 Kästchen großen Raster wird der Buchstabe entworfen.

Bild 12.6: Ein Buchstabe unter der Lupe.

Ein solches Raster nennt sich Matrix. Bei der seriellen Übertragung gibt der C 64 Punkt für Punkt dieses Rasters in einer codierten Reihenfolge durch. Bei der parallelen Datenübertragung wird der Buchstabe nicht punkt- sondern reihenweise übergeben, d.h., der C 64 übermittelt acht Punkte zugleich über je einen Kanal. Letztere Methode ist selbstverständlich schneller als die erste.

Der C 64 kennt jedoch keine parallele Datenübertragung. Die mehrspurige Autobahn für Daten muß mit einem »Interface« gebaut werden. Ha, da haben wir den merkwürdigen Begriff, der mich schon manche Überlegung gekostet hat. Ins Deutsche übersetzt bedeutet er »Zwischengesicht«, was wenig aussagt. Viel verständlicher ist aber die Umschreibung »Übersetzer für verschiedene Schnittstellen«. Das Interface ist eine Anpaßschaltung, die die einzeln ankommenden (seriellen) Daten sammelt und sie dann in entsprechenden Informationen an den Drucker weitergibt. Die Kombination Interface und Drucker bietet eine Reihe Vorteile. Das Schriftbild ist besser (NLQ-Schrift), der Ausdruck läuft wesentlich schneller und die Drucker sind auch an andere Computer anschließbar. Der Nachteil ist der Preis. Zum C 64 muß ein Interface für bis zu 300 DM angeschafft werden.

Junge, Junge, langsam aber sicher läuft der Schädel über. Diese Drucker sind eine eigene Computerwelt, neue Sprache inbegriffen. Zu diesem Zweck habe ich am Ende des Kapitels wieder ein kleines Lexikon zusammengestellt, in dem alle neuen Begriffe nachgelesen werden können. Ich brauche eine Pause, der letzte Abschnitt hat viele neue Erkenntnisse gebracht.

Eine kleine Zusammenfassung: Für den Datenverkehr zwischen dem C 64 und einem Drucker (außer den Commodore-Druckern) muß eine achtspurige Autobahn gebaut werden. Der C 64 schickt Daten an einen Drucker nur nacheinander, also eins nach dem anderen. Mit dieser Kette können die meisten Drucker nicht umgehen. Sie verarbeiten nur die Übermittlung von acht parallelen Informationen. Sie müssen auf acht Spuren nebeneinander im Drucker eintreffen. Der Ort, an dem sie in den Datenkreislauf des Druckers eindringen, nennt sich Centronics-Schnittstelle. Die achtspurige Autobahn wird durch ein Interface gebaut, das die einzeln ankommenden Daten des C 64 verarbeitet. Puh, das ist eine komplizierte Sache.

Fleißiges Interface

Sehen wir uns einige zusätzliche Funktionen des Interfaces an. Schnell taucht ein neues Problem auf. Der Drucker versteht nicht alle vom C 64 gesendeten Zeichen, weil er einen anderen Zeichensatz hat. Unter einem Zeichensatz verstehen wir die verfügbaren Zeichen. Ein Typenrad hat einen Zeichensatz, der aus den auf den Speichen befindlichen Buchstaben besteht. Wenn der Buchstabe »ü« nicht auf einer Speiche vorhanden ist, gehört er nicht zum Zeichensatz. Genauso ist das mit dem Matrix-Drucker. In verschiedenen Programmen ist uns bereits das reverse Herz zum Löschen des Bildschirms über den Weg gelaufen. Es entstand durch Drücken von [SHIFT] und [CLR/HOME]. Macht dieses Zeichen mal einem Drucker klar, der nicht von der Firma Commodore ist.

Ich kann Euch sagen, da erlebt man manch tragische Geschichte. Die ausgedruckten Zeichen stellen alles mögliche dar, nur kein reverses Herz. Hier tritt das Interface in Aktion: Es wandelt die unbekannten Zeichen (in der Fachsprache spricht man von »generieren«) des C 64 in für den Drucker verarbeitbare Informationen um, eine große Aufgabe. Wir kommen wieder auf den Begriff »Übersetzer« zurück. Das zwischengeschaltete Interface vermittelt die Daten des Computers an den Drucker. Auf diese Weise können sich die beiden verständigen.

Ich will meinem Händler nun mit genügend Fachwissen unter die Augen treten. Nochmal lasse ich mich von dessen Kauderwelsch nicht in die Flucht schlagen. Es gibt dabei viel zu bedenken.

Der Drucker ist wie gesagt ein eigener Computer. Niemand kann erwarten, daß er die gleichen Befehle versteht wie der C 64. Damit die Sache klar ist: Wir reden von Befehlen, nicht von Zeichen, die ausgedruckt werden.

Der Drucker empfängt neben auszudruckenden Zeichen sogenannte »Steuerbefehle«. Spielen wir Drucker: Wir bekommen vom C 64 Zahlenwerte über das Interface. Mit diesen Werten gehen wir zu einer großen Tabelle, in der alle Zeichen aufgelistet sind, die wir ausdrucken können. Zu jeder empfangenen Zahl gehört ein Zeichen (Bild 12.7).

Bild 12.7: Hinter jeder empfangenen Zahl verbirgt sich ein Zeichen.

Wir nehmen die benötigten Zeichen aus der Tabelle und drucken sie aus. Nach einer Weile kommen wir am Ende der Zeile auf dem Papier an. Was tun, wir können doch nicht einfach über das Papier hinausschreiben. Ein Steuerbefehl fehlt. Dieser sagt uns: »Hier ist die Zeile zu Ende, auf dem Papier ist kein Platz mehr vorhanden. Deswegen mußt du an den Anfang der nächsten Zeile gehen und dort weiterdrucken«. Im Verlauf eines solchen Ausdrucks (zum Beispiel eines Textes) kommt eine ganze Reihe von Steuerbefehlen vor. Wenn Ihr zum Beispiel eine Überschrift in dicken schwarzen Buchstaben drucken wollt, muß der Drucker vorher den entsprechenden Steuerbefehl für »Fettschrift« erhalten. Diese Befehle waren lange Zeit von Drucker zu Drucker verschieden und es gab immer wieder Probleme beim Ausdruck. Stellt Euch folgenden Fall vor: Euer Textverarbeitungsprogramm arbeitet mit dem neuen Drucker nicht zusammen, weil der Drucker die Steuerbefehle nicht versteht. Eine traurige Kombination, die aber leider oft harte Realität ist. Die einzige Lösung wäre, das Programm so umzuschreiben, daß es auf dem neuen Drucker läuft. Wir müßten die alten Steuerbefehle gegen neue Befehle austauschen. Keine einfache Aufgabe. Seit einiger Zeit gibt es eine andere Lösung: den ESC/P-Standard.

Drucker unter sich

ESC/P bedeutet »Epson Standard Codes for Printers« und das heißt übersetzt: »Allgemeiner Befehlssatz für Epson-Drucker«. Das Wort »Epson« ist ein Firmenname. Der Drucker-Hersteller »Epson« hat einen von den meisten Produzenten akzeptierten Steuerbefehlssatz erarbeitet. Das bedeutet: Alle Drucker mit ESC/P haben die gleichen Steuerbefehle. Ein Textverarbeitungsprogramm, das mit einem Epson-Drucker problemlos läuft, funktioniert auf allen anderen Druckern mit ESC/P. Die Anpassung des Druckers an Computer und Programm ist erheblich einfacher.

In der Fachsprache spricht man auch von Epson-kompatibel. Ist ein Drucker Epson-kompatibel, arbeitet er mit den gleichen Befehlen wie ein Epson-Drucker. Der betreffende Drucker ist dann mit einem Epson-Drucker austauschbar (weil er auch mit ESC/P arbeitet).

Puh, langsam sinke ich zu Boden. Rauchwolken steigen aus meinem Kopf empor. Mühsam schleppe ich mich bis zum Kühlschrank und lösche den Brand mit einer Cola. Nach einer Weile kann ich aufstehen und gehe wieder an die Arbeit. Mit einer kleinen Zusammenfassung der neuesten Erkenntnisse verschaffe ich mir einen besseren Überblick. Ein auf dem Computer geschriebener Text soll ausgedruckt werden. Für den Ausdruck benötigt der Drucker neben dem auszudruckenden Text Steuerbefehle: Wie lang soll eine Zeile auf dem Papier sein, wieviel Abstand soll zwischen den einzelnen Zeilen sein, welche Schriftart wird verwendet? Wenn die Steuerbefehle in »Epson Standard Codes for Printers« gesendet werden, so läuft das Programm problemlos auf fast allen Druckern (Ausnahmen gibt es immer) von Epson und Epson-kompatiblen Geräten. Wenn ein Drucker kompatibel ist, so steht das auf einem kleinen Schild am Gehäuse des Druckers oder im Bedienungshandbuch. Der Ausdruck »Kompatibilität« oder »kompatibel« findet auch in anderen Computer-Bereichen Anwendung. Zwei Computer sind zum Beispiel kompatibel, wenn alle Programme eines Computers auch auf dem anderen laufen. Alle Basic-Programme des C 64 funktionieren auch auf dem C 128, solange sich keine POKE- oder PEEK-Befehle darin befinden (s. Kapitel 14). Der Commodore 64 ist zum Commodore 128 »kompatibel«.

Der Weg aus dem Urwald

Jetzt starte ich meinen dritten Drucker-Versuch. Ich nehme mein Textverarbeitungsprogramm, einen mit diesem Programm geschriebenen Text auf Diskette und gehe zum altbekannten Händler. Der sieht mich kommen und freut sich schon darauf, wie er mir mit seinem Fachchinesisch imponieren kann. Nicht mit mir! Ich knalle ihm meine Diskette auf den Tisch und sage: »Bruder, ich will einen Epson-kompatiblen Drucker mit Interface für meinen C 64 kaufen. Hier ist mein Lieblings-Textprogramm und ein Beispiel-Text. Machen wir einen kleinen Probeausdruck, okay?«

Der Verkäufer ist verunsichert, so kennt er mich nicht. Zuerst will er mich abspeisen: »Das probieren Sie dann am besten zu Hause aus«. Ich bleibe hart und bestehe auf einer Probevorführung, schließlich lasse ich vielleicht eine Menge Geld in diesem Laden und Kundendienst wird großgeschrieben!

Das haben wir gelernt

Centronics-Schnittstelle: Viele moderne Drucker sind mit einer Centronics-Schnittstelle ausgerüstet. Sie ist das Eingangstor für die vom Computer ankommenden Daten und Informationen. Eine Centronics-Schnittstelle verarbeitet nur parallel gesendete Informationen.

ESC/P: Bedeutet »Epson Standard Codes for Printers« und bezeichnet einen festgelegten Befehlssatz für die Steuerung von Druckern. Dieser Standard vereinfacht die Anpassung von Druckern an Programme, da die Drucker mit den gleichen Steuerbefehlen arbeiten.

Grafikfähigkeit: Die Grafikfähigkeit ist die Fähigkeit eines Druckers zu Darstellung und Ausdruck von Computergrafiken. Je höher die Grafikfähigkeit, desto besser sind die ausgedruckten Grafiken.

Interface: Ein Interface ist eine Anpaßschaltung für die Verbindung zweier miteinander in Beziehung stehender Systeme. Die Tastatur ist zum Beispiel die Schnittstelle vom C 64 zu uns. In unserem Fall für den Datenaustausch zwischen C 64 und Drucker. Es wandelt die seriell aus dem C 64 kommenden Daten in parallele und gibt sie an den Drucker weiter.

Kompatibilität: Wörtlich »Verträglichkeit«. Ein Computer ist zu einem anderen kompatibel, wenn seine Programme auch auf einem Computer fremder Bauart funktionieren. Im Drucker-Bereich wird häufig der Begriff »Epson-kompatibel« verwendet.

Matrix-Drucker: Drucker, der die Zeichen mit Nadeln als Punktraster druckt. Diese Nadeln sind in einer bestimmten Anordnung (Matrix) gruppiert. Der Buchstabe wird auf das Papier »gepiekst«. Es gibt 7-, 9-, 18- und 24-Nadel-Matrix-Drucker.

NLQ-Schrift: NLQ bedeutet »Near-Letter-Quality« (»Fast-Brief-Qualität«). Ein Drucker mit NLQ-Funktion liefert Ausdrucke, die von der Qualität her fast das Schriftbild einer Schreibmaschine erreichen.

Parallele Datenübertragung: Übertragung eines Datenzeichens in einem Schritt. Sämtliche zur Darstellung des Zeichens nötigen Informationen werden zeitgleich über ebensoviele Datenleitungen transportiert.

Schnittstelle: Der Ort, an dem in den Datenkreislauf eines Computers eingegriffen wird. An einer Schnittstelle können Informationen entnommen oder in den Computer eingegeben werden.

Serielle Datenübertragung: Ausgabe einzelner Informationen, die hintereinander (in Serie) aus dem Computer »herauströpfeln«. Es kann immer nur ein Zeichen ausgegeben werden, dann kommt das nächste. Die serielle Datenübertragung kann mit einer einspurigen Autobahn verglichen werden.

Kapitel 13

Hausputz beim C 64 und was im Notfall zu tun ist

Wir haben eine Menge Spaß mit dem C 64. Er ist ein guter Freund geworden und Freundschaften muß man pflegen! Diese Pflege fordert von uns ein paar Kleinigkeiten. Nützliche Arbeiten, die das Leben des C 64 verlängern. Das Stichwort heißt »Gerätepflege«. Unser Computer soll uns so lange wie möglich erhalten bleiben.

Der größte Feind des Computers ist der Staub. Staub besteht im Grunde aus winzigen Fusseln, die alleine ungefährlich sind. Jeder kennt die unangenehme Eigenschaft von Staub: er rottet sich zusammen und fällt in Form mehr oder weniger großer Flocken über unsere Wohnungen her. Ganz besonders schlimm sind elektrische Geräte dran. Der Staub bahnt sich seinen Weg durch Lüftungsschlitze oder sonstige Öffnungen, immer mit dem Ziel, das Gerät lahmzulegen. Stellt Euch folgendes Bild vor: Eine Staubflocke legt sich über das Innere des Computers, und mit einem heftigen Britzeln verabschiedet sich der heißgeliebte Plastikkasten. Ganz so schlimm ist es in Wirklichkeit nicht, aber Schmutz kann zu Funktionsstörungen führen. Ihr lebt ein glückliches Computer-Leben, der einzige Feind ist beständig auf Euch herabregnender Dreck. Nicht mit uns. Der C 64 sollte zum Beispiel mit einer Abdeckhaube aus Kunstoff geschützt werden, wenn er gerade nicht benutzt wird.

Ganz sorgfältige Menschen packen ihn nach jeder Anwendung in eine Samthülle. Besondere Aufmerksamkeit gilt der Tastatur, dem Verbindungsmann zwischen uns und dem Computer.

Gut gepinselt ist halb gesäubert

Während wir Computerspiele spielen oder uns den Kopf über Programmier-Probleme zerbrechen, rieselt eine Menge Dreck auf den Computer. Das läßt sich nicht verhindern. Elektrische Geräte ziehen herumfliegenden Dreck besonders gut an, da sie ein magnetisches Feld um sich aufbauen, wenn Strom in ihnen fließt. Besonders die Tastatur ist beliebter Landeplatz für Müll aller Art. Klar, sie liegt in voller Breite oben

»auf« dem C 64 und ist somit ungedeckte Zielscheibe für Schmutz. Ein gefährliches und trotzdem beliebtes Spiel ist das Rauchen und Kaffeetrinken beim Computern. Es gibt Geschichten von heldenhaften C 64, die eine über die Tastatur verschüttete Tasse Kaffee verkraftet haben. Wir sollten es nicht darauf ankommen lassen: Essen und Trinken bitte nicht über dem Computer!

In Geschäften finden wir viele Reinigungsmittel für Computer in den Regalen. Es gibt spezielle Saubermacher für Bildschirme, für Gehäuse, für Disketten-Stationen und so weiter. Viele von ihnen sind aggressive Reiniger, die dem Schmutz ganz hervorragend ans Leder gehen, meistens leider auch der Umwelt und nicht selten dem Menschen direkt. Wir liegen im Trend der Zeit und verbannen solche Mittel aus unserer Reichweite. Erstens muß unser C 64 nicht weißer als weiß sein und zweitens sind die meisten Reiniger für Computer überteuert und zudem unnötig.

Wie schon gesagt, muß die Tastatur den meisten Dreck auffangen. Nun, die Computer-Hersteller sind natürlich nicht blöd und haben vorgesorgt. Der feine Staub, der sich zwischen die Tasten zwängt, fällt nicht in den C 64. Zwischen Tastatur und Innenleben des Computers liegt eine Platte. Sie fängt fast alles auf. Die Gefahr, daß es hier zu Kurzschlüssen oder Funktionsstörungen durch Staub kommt, ist relativ gering. Es ist schon vielmehr das Auge, das von verschmierten Tasten und den staubigen Tastaturen beleidigt wird. Die Reinigung der Tastatur sollte regelmäßig vollzogen werden. Vorher den Computer ausschalten und den Netzstecker ziehen. Mit einem feinen Pinsel gehen wir zwischen die Tasten und holen so den Staub aus den Ritzen. Einfach, aber gut. In der Regel reicht das aus. Wer ganz genau sein will, kann auch kleine Staubsauger benutzen. Sie gibt es in manchen Fotofachgeschäften oder Versandhäusern für Computerzubehör.

Etwas komplizierter wird es, wenn wir Limonade oder gezuckerten Kaffee über die Tastatur verteilen. Selbiges gilt für arg verschmutzte Tastaturen. Hier hilft nur eins: alle Tasten abziehen und säubern. Und das ist kein Scherz. Getrocknete Limonade kann nämlich das bewirken, was wir vom Schmutz nicht zu fürchten brauchen – es treten Funktionsstörungen auf. Da Limo sich durch die engsten Ritzen zwängt, können zwei Dinge geschehen.

1. Die Taste verklebt und läßt sich nur noch mit Gewalt drücken. Hat man es dann geschafft, bleibt sie meistens »gedrückt«. Mit ihr kann nicht mehr vernünftig gearbeitet werden.

2. Die Limonade schafft eine Brücke zwischen zwei Leitungen. Die Tastatur wird dadurch zwar nicht beschädigt, es kann aber zum Beispiel passieren, daß wir ein »A« drücken und auf dem Monitor erscheint ein »Q«.

Um an die betreffenden Stellen ranzukommen, müssen wir alle Tasten von der Tastatur abziehen. Die Tasten des C 64 sind nämlich lediglich auf einen Stift gesteckt (Bild 13.1). Das bedeutet, wir können sie nach Belieben abziehen und wieder aufstecken.

Aber Vorsicht, zwei Dinge müssen beachtet werden: Jede Taste hat eine Feder. Sie schiebt die Taste wieder zurück, nachdem sie von uns gedrückt wurde. Ohne die vollständige Anzahl Federn gibt es Probleme – die Tastatur funktioniert dann nach der Reinigung noch schlechter, und versucht mal eine solche Feder einzeln zu bekommen. Also deshalb gut aufbewahren. Auf der anderen Seite solltet Ihr sicher sein, daß noch eine Tabelle der Tastatur vorhanden ist. Schließlich müssen die Tasten in richtiger Anordnung wieder aufgesteckt werden.

Bild 13.1: Die Tasten des C 64 sind auf Stifte aufgesteckt.

Zum Abziehen der Tasten gibt es zwei Möglichkeiten. In Elektronik-Geschäften gibt es einen Tastenabzieher zu kaufen. Mit ihm geht alles ganz einfach. Fast ebensogut funktioniert dieser Vorgang mit zwei stinknormalen Feinmechaniker-Schraubendrehern. Einfach an zwei gegenüberliegenden Stellen anwinkeln und vorsichtig rausziehen. Vorsicht ist geboten, weil der Stift, auf dem die Taste hängt, bei roher Gewalt abbrechen kann und die Taste zerkratzen kann, wenn wir abrutschen. Sind endlich alle Tasten ab, beginnt die Putzerei.

Mit einem Fön befreien wir den C 64 vom Staub. Ein Glas Wasser mit ein paar Tropfen Brennspiritus halten wir natürlich parat. Ein fusselfreies Tuch um den geeignetsten Finger wickeln, in das Glas Wasser eintauchen und anschließend damit über die freie Fläche reiben. Ich verspreche Euch, das ist der reinste Limonaden-Killer. In der Regel reicht diese Methode auch bei verklebten Tastenstiften. Wenn nicht, müßt Ihr einen Tropfen Brennspiritus auf den Stift träufeln und ihn dann so oft drücken und wieder rausziehen, bis es ganz leicht geht. Ihr werdet nie wieder Probleme mit dieser Taste

haben – bis zum nächsten Glas Limo. Das Zusammenbauen dürfte wohl klar sein, vergeßt nur nicht die Federn.

Der Monitor ist ein noch größerer Staub-Freund als die Tastatur. Auf dem Bildschirm setzt sich im Laufe der Zeit ein Staubfilm ab. Dies geschieht aus schon erwähnter elektromagnetischer Anziehung. Schmutz auf der Monitorscheibe stört immens, da er im wahrsten Sinne des Wortes direkt ins Auge sticht. Er muß entfernt werden. Vor der Reinigung des Bildschirms sollte der Monitor ausgeschaltet sein, noch besser: Netzstecker ziehen. Außerdem sollte er zuvor mehrere Stunden unbenutzt gewesen sein. Auf dem Bildschirm kann unter Umständen eine Oberflächenspannung von mehreren tausend Volt liegen, die Ihr ganz schön zu spüren bekommt. Da die anliegende Stromstärke nur im Milli-Ampere-Bereich liegt, ist sie in der Regel nicht tödlich, es sei denn, Ihr habt ein schwaches Herz. Dennoch ist sie unangenehm.

Staub wird mit einem trockenen Lappen, der langsam und vorsichtig über den Bildschirm geführt wird, entfernt. Fettige Fingerabdrücke bekämpft lauwarmes Wasser mit ein paar Tropfen Brennspiritus. Es kann allerdings sein, daß wir mit unseren Fingern nicht nur fettige Abdrücke hinterlassen haben, sondern auch mit unseren Griffeln Marmelade auf den Bildschirm übertragen haben. In diesem Fall ist unser umweltfreundliches Hausmittel, lauwarmes Wasser mit Brennspritus, relativ unwirksam. In solchen Fällen helfen einige Tröpfchen Allzweckreiniger in unserem Wasser. Wer auf kommerzielle Bildschirmreiniger nicht verzichten will, sollte auf normale Glasreiniger zurückgreifen. »Spezielle« Bildschirmreiniger für Computer sind im Prinzip auch nichts anderes, nur eben teurer.

Leute, tut Euch und mir den Gefallen und laßt die Finger vom Innenleben des Monitors oder Fernsehers. Wer hier herumfummelt, setzt sich wegen der hohen Spannung in den Geräten großen Gefahren aus. Im Reparaturfall hat der Fachmann das Sagen, auch wenn das Portemonnaie einen Weinkrampf kriegt, okay? Nur ein Tip. Durch die vielen Belüftungsschlitze rieselt der Staub mit Freude durch. Es kann passieren, daß sich Staub irgendwo sammelt und einen Kurzschluß erzeugt. Stellt deshalb den Monitor in einem relativ staubfreien Raum auf und deckt ihn bei Nichtgebrauch ab. Hierzu reicht ein normales Tuch, es gibt aber auch spezielle Abdeckhauben für Commodore-Monitore.

Fusseln und Staub bedrohen auch Disketten-Station und Datasette. Der Tonkopf der Datasette muß von Zeit zu Zeit gereinigt werden. Zu diesem Zweck besorgt Ihr Euch reinen Alkohol (Methanol oder Isopropanol) und aus der Drogerie Wattestäbchen für die Ohren. Diese tränkt Ihr mit dem Alkohol. Jetzt drückt Ihr bei geöffneter Kassetten-Lade den PLAY-Schalter: Zwei kleine magnetische Vierecke schieben sich nach vorne. Im rechten befindet sich der Tonkopf zur Wiedergabe, im linken der zur Aufnahme. Reibt vorsichtig mit dem Wattestäbchen über die Köpfe.

Reibt nicht zu fest, denn sonst können die Tonköpfe zerkratzen und wir haben genau das Gegenteil von dem erreicht, was wir wollten. Schaltet nun den C 64 ein – die Datasette arbeitet. Ganz rechts unten seht Ihr ein sich drehendes Gummirad. Dieses muß auch sauber sein. Haltet einmal das feuchte Wattestäbchen daran. Ihr werdet staunen, wieviel Dreck sich dort angesammelt hat. Wahrscheinlich reicht ein Stäbchen nicht aus.

Keine Reinigung am Vliesband

Leute, macht bloß nicht den Fehler und geht in ein Geschäft, um dort Reinigungskassetten zu kaufen. Diese sind sehr umstritten. Die einen schwören drauf, Leute wie wir verbannen sie. Die Technik solcher Reinigungskassetten basiert auf dem Naß-Trocken-Verfahren. Ein Vlies in Form eines Tonbandes wird mit Alkohol beträufelt. Es reibt über die Tonköpfe, dabei reinigt es mit den nassen Stellen und trocknet mit den trockenen Vliesteilen. Im Prinzip nichts anderes als unsere Wattestäbchen-Technik, eben nur kostenintensiver.

Was für die Datasette gilt, muß nicht immer für Disketten-Stationen zählen. Für sie hat sich das Naß-Trocken-Verfahren bewährt. Ein verdreckter Schreib-Lese-Kopf des Disketten-Laufwerks kann nicht richtig arbeiten: er schreibt fehlerhafte Daten oder liest sie nicht richtig von Diskette, eine üble Vorstellung. Hier ist die Reinigung per Hand zwar ebenfalls günstiger als kommerzielle Lösungen, steht jedoch in keiner Relation zum Aufwand. Eine Reinigungs-Diskette muß her. Im Verhältnis zum Lieferumfang sind sie zwar unverschämt teuer, dafür aber nützlich. In ihr befindet sich eine vliesartig beschichtete Diskette. Das heißt: Anstelle einer Diskette befindet sich ein Reinigungsmaterial in der Schutzhülle, das durch Drehen den Lese-Kopf von allen Ablagerungen befreit. Die Reinigung des Lese-Kopfes sollte vor seinem Ausfall geschehen! Fangt früh genug an!

Was tun, wenn es zu spät ist?

Jetzt kommt ein trauriges Thema, mit dem wir uns am liebsten gar nicht befassen würden. Was tun, wenn der treue Freund und Begleiter plötzlich seinen Dienst versagt? Was tun, wenn der C 64 defekt ist?

Die Hitzköpfigen unter uns nehmen den heißgeliebten Computer unter den Arm und laufen direkt zum nächsten Händler. Das muß nicht sein. Erstmal einen Totalcheck machen.

1. Sind alle Stecker richtig beziehungsweise überhaupt eingesteckt?
2. Liegt der Fehler wirklich am C 64 oder an der Software? Also im Zweifelsfalle den C 64 aus- und dann wieder einschalten und ein anderes Programm starten.

3. Häufig kommt es auch zu Fehlfunktionen, wenn angeschlossene Geräte defekt sind. Ein Beispiel: Ist ein Joystick mit einem Kurzschluß mit dem C 64 verbunden, funktioniert unter Umständen die Tastatur nicht mehr richtig. Ein »Defekt«, der oft vorkommt. Noch trivialer ist, wenn der Joystick zwar in Ordnung ist, das Dauerfeuer jedoch die Tastatur stört. Also erstmal alle unnötigen Kabel aus dem C 64 rausziehen. Und dann nochmal versuchen.

Ist der Defekt nun immer noch nicht behoben, bleibt uns leider nichts anderes übrig: Wir müssen zur Fachwerkstatt. Diese sind meistens unverschämt teuer. Nach dem Motto, entweder man hat das Wissen oder man muß dafür zahlen (und das ordentlich), dürfen wir blechen. Wir sind ja nur einfache Anwender. Ihr alle wißt, was Euch der C 64 gekostet hat. Muß er wirklich zur Reparatur, laßt Euch vorher einen Kostenvoranschlag machen. Die Werkstatt sagt damit, wieviel die Reparatur ungefähr kosten wird. Anhand dieser Information könnt Ihr entscheiden, ob sich eine Reparatur lohnt oder eher ein Neukauf angesagt ist.

Einige werden vielleicht hören: »Die Reparatur wird den Wert des C 64 übersteigen«. Na gut, wir kaufen einen neuen. Aber was geschieht mit dem alten? In Computer-Zeitschriften gibt es massenhaft Kleinanzeigen mit folgendem oder ähnlichem Wortlaut: »Kaufe jeden C 64, auch defekt. Zahle Höchstpreise.« Die Verlockung ist groß. Ein Anruf bestätigt: Für meine kaputte Möhre kriege ich noch 70 Piepen. Doch solchen Kleinanzeigen sollten wir genauso vertrauen wie Autohändlern, die Schilder aushängen oder stehen haben wie: »Kaufe jedes Auto – Barauszahlung«, nämlich garnicht. Da wir unseren »Geschäftspartner« nicht kennen, sollten wir die Finger davon lassen, sonst geht es uns wie einem Bekannten von mir. Der hat seinen C 64 an so einen »lieben« Menschen geschickt. Geld hat er nie bekommen, dafür kam sein C 64 per Post wieder, der letzten funktionierenden Teile beraubt. Da mein Freund diesen Betrug nicht beweisen konnte, war er machtlos. Leser des 64'er-Magazins haben ähnliches geschildert.

Bleibt das Problem, was mit dem defekten C 64 zu tun ist. Wenn Ihr einen neuen Computer kaufen wollt, bietet den alten der Werkstatt an. Diese machen genau dasselbe wie die Leute mit den Kleinanzeigen – sie schlachten ihn aus, um funktionierende Teile in defekte 64er einzubauen. Nach dem Motto: Aus zwei defekten machen wir einen funktionierenden. Die Werkstatt verspricht Euch wohl kaum 70 Scheine für diese Krücke, aber immerhin ein paar Mark. Wenn sie selbst das nicht tut, springen sicher ein paar Prozente Rabatt bei einem Neukauf raus. Einfach mal fragen.

Ansonsten kann ich nur raten, den defekten C 64 zu behalten. Irgendwann, wenn wir mal fit sind, können diese oder jene Teile wieder verwendet werden. Man weiß ja nie. Na ja, bisher läuft der Computer ja einwandfrei. Wenden wir uns erfreulicheren Dingen zu. Auf ins nächste Kapitel.

Das haben wir gelernt

Pflege: Obwohl der C 64 nur ein »Ding« ist, braucht er schon mal Pflege. Er muß gesäubert werden.

Reinigungsmittel: Werden aus unserer Reichweite verbannt. Zu unserem Pflegeset gehört ein Pinsel, Brennspiritus, lauwarmes Wasser, ein fusselfreies Tuch und Wattestäbchen. Herkömmliche Glasreiniger sind unerwünscht, aber, fein dosiert, vertretbar.

Totalschaden: Muß nicht sein, auch wenn es ganz schlimm aussieht. Deshalb erst mal checken. Wenn doch, dann ab zur nächsten Reparatur-Werkstatt und dort einen Kostenvoranschlag verlangen. Erst danach entscheiden, ob Reparatur oder Neukauf angesagt ist.

Kapitel 14

Homo ludens oder die Lust am Spielen

Programmieren ist stark, aber so ab und zu habe ich davon die Nase voll. Leute, wir müssen mal entspannen. Wie wär's mit einem Spiel. Unser C 64 ist nämlich nicht nur bekannt wegen seiner guten Grafik und tollen Musik, die man mit ihm machen kann, sondern in erster Linie wegen der vielen, vielen Spiele, die es für ihn gibt. Mittlerweile gibt es so viele für den C 64, daß wir die Spiele in verschiedene Typen untergliedern müssen. Ihr werdet feststellen, daß Ihr die eine Spielart mehr mögt als eine andere.

Als ich mein erstes Spiel kaufte, ging das ungefähr so: Rein in den Laden, ein Spiel mit einer schönen Hülle und reißerischer Beschreibung gekauft, zuhause geladen und bitterböse enttäuscht gewesen. Nun, das Spiel an sich war nicht schlecht, aber es war nicht mein Fall, auch wenn die Beschreibung interessant klang.

Deshalb überlegen wir uns erst mal, was wir unter einem tollen Spiel verstehen. Ich unterteile in fünf verschiedene Spieltypen:

1. Sportspiele
2. Adventures
3. Schießspiele
4. Geschicklichkeitsspiele
5. Simulationen

Die Numerierung hat nichts Besonderes zu bedeuten. Es ist lediglich meine Reihenfolge, nach Geschmack geordnet, die bei Euch natürlich anders aussehen darf.

Fangen wir mit den Simulationen an. Simulation ist im normalen Leben die bewußte Vortäuschung von Krankheiten oder die übertriebene Darstellung ihrer Symptome durch den Simulant. Der Schüler, der sich krank stellt, weil er eine Klassenarbeit nicht mitschreiben will, simuliert eine Krankheit. Er ist ein Simulant. Simulation im wissenschaftlichen Sinne bedeutet die Nachbildung oder Darstellung von physikalischen, technischen und auch biologischen Vorgängen. Anhand mathematischer Modelle wird eine

wirklichkeitsnahe Untersuchung dieser Prozesse ermöglicht. Sie sind meist preiswerter und ungefährlicher als die am Original.

Flugsimulatoren ermöglichen dem Flugschüler ein gefahrloses Üben von Landen und Starten, bevor er sich in eine richtige Maschine setzt. Die Simulationen für den C 64 sind natürlich nicht so realitätsnah wie die der NASA oder Lufthansa, dennoch machen sie Spaß, wenn man sich dafür interessiert. Für den C 64 gibt es hauptsächlich Flug- und Fahrsimulationen. Zwei Vertreter sind »ThunderChopper« (Bild 14.1, die Bilder zu den Spielen findet Ihr am Ende des Kapitels) und »Test drive« (Bild 14.2). In beiden Simulationen sind wir die Piloten, die eine bestimmte Strecke abfliegen oder fahren müssen, und dabei oft vor schwierige Situationen gestellt werden. Rasch werden wir zu Bruchpiloten, ich schwöre es Euch. Mit Wirtschaftssimulationen wie »Airline« (Bild 14.3) werden wir mit ein wenig Geschick zu Jungmanagern. Hier geht es nicht um Fahr- oder Flugtauglichkeit, sondern um kühle Köpfe und heiße Finanzen. Wir haben die Verantwortung über Personal, Flugnetz, Service und viele andere Dinge, wie zum Beispiel den Kauf neuer Flugzeuge. Ähnliche Simulationen gibt es sehr häufig.

Geschicklichkeitsspiele verlangen uns meist viel Joystick-Gefühl ab. Hier geht es im Prinzip darum, irgend einen Gegenstand über fallengespickte Wege und Gänge ans Ziel zu bringen. »Block' n' Bubble« (Bild 14.4) aus dem 64'er Sonderheft 26 ist ein typischer und guter Vertreter dieser Spieleklasse. Unser Demo-Spiel auf der Diskette zum Buch »The Great Giana Sisters« ist ebenfalls ein Geschicklichkeitsspiel. Hier muß in richtiger Weise gesprungen und gehandelt werden, um ans Ziel zu kommen. Und obwohl Giana auch mal schießt, zählt es nicht zu den Schießspielen.

Diese sind nämlich Spiele, in denen es nur um das eine geht – ums Schießen. Bei dieser Spieleart merkt man schnell, daß es ziemlich egal ist, welches Spiel sich gerade im Speicher des C 64 befindet. Rasch ist einem die ewige Ballerei über. Ziel ist es meistens, so viele Gegner verschiedenster Gemeinheit wie möglich zu vernichten, um so die Welt vor den bösen Feinden zu retten. Typischer Vertreter ist hier »IO« (Bild 14.5). Es zeichnet sich durch hervorragende Grafik aus, entbehrt aber jeglichen Spielsinns. Viele finden es dennoch toll. Zugegeben, wenn mir der Kopf vor lauter Basic qualmt, ziehe ich solche Spiele vor.

An dieser Stelle muß gesagt werden, daß es die Typen Geschicklichkeitsspiele und Schießspiele kaum in »chemisch« reiner Form gibt. Jedes enthält Elemente des anderen. So könnten die Programmierer von »IO« durchaus behaupten, es sei ein Geschicklichkeitsspiel mit viel Ballerei. Es ist also oft subjektiv, in welche Sparte man ein Spiel einteilt.

Wird bei Schießspielen ein Spielsinn oft vermißt, so sind Adventures die krassen Gegenteile. Hier ist der Spieler meistens ein Held ohne Furcht und Tadel, der das Land vom Bösen befreien oder die entführte Prinzessin finden soll. Das führt den wackeren Krieger in der Regel in dunkle Grotten, in denen magische Gegner schon auf

sein Erscheinen warten. Bei Adventures müssen wir unseren Kopf ziemlich anstrengen, denn oft geht es nur weiter, wenn wir ein Rätsel oder eine Aufgabe gelöst haben.

Bei Adventures unterscheiden wir zwischen Text-, Grafik- und Action-Adventures. Text-Adventures bestehen nur aus Text, das heißt, der C 64 teilt uns schriftlich mit, in welcher Gegend wir uns befinden, was wir sehen und in welche Richtungen wir weitergehen können. Ungefähr so: »Sie stehen mitten im Wald. Es ist düster. Im Osten sehen Sie einen verlassenen Friedhof. Aus Süden hören Sie Stimmen. Es gibt Ausgänge nach Norden, Süden und Osten.« Mit Ausgängen sind natürlich die Wege gemeint, durch die die aktuelle Spielposition verlassen werden kann. Reiner Text strapaziert unsere Vorstellungskraft ganz schön. Grafik-Adventures machen es uns da einfacher. Schöne Bilder zeigen die Landschaft oder die Gruft, in der wir stehen. Manchmal zeigen sie Bilder der Personen oder Gegenstände, mit denen wir zu tun haben. Auch hier teilt der C 64 mit, was wir sehen und in welche Richtung wir weitergehen können. Anders ist es bei Action-Adventures. Bei ihnen können wir unsere Spielfigur in der Regel mit dem Joystick über den Bildschirm durch Gänge, Gassen und Grotten führen. Wir können Gegner mit einer Waffe bekämpfen oder Zaubersprüche schleudern, so wie bei »Demon Stalker« (Bild 14.6). In diesem Spiel müssen wir in jeder Spielebene eine besondere Aufgabe lösen, zum Beispiel alle Schatztruhen finden oder alle Treppen benutzen.

Meine Lieblingsspiele sind hingegen Sportspiele oder auch – ich höre es nicht so gerne – Sportsimulationen. Man sollte nicht etwa dem Glauben verfallen, der Spieler (also wir) wird hier zum Sport verleitet. Sport treibt in erster Linie die Spielfigur. Wir müssen sie lediglich dazu veranlassen. Dies geschieht in 99 Prozent aller Fälle durch den Joystick. Wie bei allen Spieletypen gibt es auch in dieser Sparte gute und miese Vertreter. Unter miesen Vertretern verstehe ich Sportspiele, bei denen es nur darauf ankommt, den Joystick so schnell wie möglich von links nach rechts zu rütteln. Bei diesen sind es tatsächlich wir, die schwer ins Schwitzen kommen. Solche Rüttelspiele waren die ersten Sportspiele, sind aber so gut wie ausgestorben. Heute finden wir eher Spiele, bei denen es auf gefühlvolle oder geschickte Joystick-Führung ankommt. »Skate or die« ist eins dieser Art, es ist neben einem Sport- auch ein Geschicklichkeitsspiel.

Die Ausrüstung

Ihr werdet es schon gemerkt haben, unverzichtbares Werkzeug für professionelle Spieler ist der Joystick. Tatsächlich liegt es an der Wahl dieses kleinen Gerätes, ob Ihr gute oder schlechte Ergebnisse erzielt. Gute Joysticks haben sechs Mikroschalter und Dauerfeuer. Der Competition Pro Extra (Bild 14.7) wurde vom 64'er-Magazin zum Referenz-Joystick gekrönt, und die Redakteure dort wissen, wie man Joysticks testet. Aber auch andere Joysticks sind klasse. Geht doch einfach in ein Geschäft und probiert

verschiedene aus. Nehmt nur den mit nach Hause, der Euch am besten zusagt. Bei der Kaufentscheidung sollten jedoch grundsätzliche Merkmale eine Rolle spielen.

Bild 14.7: Ein hervorragender Joystick: der Competition Pro Extra.

Wir unterteilen einen Joystick in sichtbare und vorsichtbare Kennzeichen. Sichtbar ist zunächst alles, was mit bloßem Auge erkennbar oder durch Hilfsmittel (z.B. ein Lineal) nachweisbar ist. Vorsichtbar sind Dinge, die erstmal nicht zu erkennen sind, mittels Tricks jedoch ans Tageslicht gebracht werden können, zum Beispiel durch Öffnen des Gehäuses. Wie das Wort »vorsichtbar« schon sagt, gilt diesem Bereich besondere Aufmerksamkeit.

Die sichtbaren Merkmale

Ein durchschnittlicher Joystick sollte ein Kabel mit einer Länge von mindestens 1,3 Metern haben. Dauerfeuer darf nicht fehlen und vier Saugnäpfe zur Befestigung am Tisch müssen vorhanden sein. Wichtig ist, daß der Griff nicht zu labbrig in der Führung liegt. Er muß ganz straff in die acht Richtungen zu bewegen sein, wobei der Weg von links nach rechts beziehungsweise von vorne nach hinten weder zu groß noch zu klein sein darf.

Letzteres müßt Ihr selbst beurteilen. Nehmt den Joystick in die Hand und prüft, ob er gut zu halten ist, kommt Ihr auch an den Feuerknopf, ohne einen Krampf zu kriegen?

Die vorsichtbaren Merkmale

Vorsichtbar nennen wir Dinge oder Eigenschaften, die auf den ersten Blick unsichtbar sind, jedoch sichtbar gemacht werden können. Der Motor eines Autos ist zum Beispiel vorsichtbar. Normalerweise können wir ihn nicht sehen, das ändert sich mit Öffnen der Motorhaube. Analog zum Auto öffnen wir beim Joystick das Gehäuse. Interessant für uns sind die im Inneren befindlichen Schalter. Ein guter Joystick hat mindestens sechs Mikroschalter. Nun, der Verkäufer wird uns was erzählen, wenn wir in seinem Geschäft so einfach einen Joystick aufschrauben. Das brauchen wir auch nicht. Hat ein Joystick Mikroschalter, so steht das meist auf der Verpackung. Finden wir auf der Verpackung also nirgends den Hinweis auf Mikroschalter, lassen wir ihn am besten gleich im Regal. Wenn wir zwei oder drei Joysticks in die engere Wahl gezogen haben, werden wir zum Verkäufer gehen und ihn bitten, ein Spiel für uns in den C 64 zu laden. So testen wir den Joystick in der Praxis. Ein cleverer Joystick-Kauf braucht Zeit. Enttäuschungen lassen sich vermeiden. Vor allem die Liste nicht vergessen.

Mit einem Joystick gut ausgerüstet geht es nun ans Werk. Probiert einmal selbst aus, welche Spieletypen Euch am besten gefallen. Ein kleiner Tip zu »The Great Giana Sisters«: Im richtigen Spiel (also nicht im Demo) springt Ihr unbeschädigt in den nächsten Level, wenn Ihr zugleich die folgenden Tasten drückt: [A] [R] [M] [I] [N]. Viele Tips zu Spielen findet Ihr in der Zeitschrift »Happy Computer«. Also dann: Keep on playing – bis zum nächsten Kapitel.

Das haben wir gelernt

Spielearten: Es gibt eine ganze Reihe von verschiedenen Spielen. Sie können grob in fünf Klassen eingeteilt werden:

1. Sportspiele
2. Adventures
3. Schießspiele
4. Geschicklichkeitsspiele
5. Simulationen

Joystick-Kauf: Der Joystick-Kauf sollte sorgfältig geplant werden. Vom Joystick hängt in hohem Maße der Spaß ab. Ein schlabberiger, ausgeleierter Joystick vermiest selbst das tollste Spiel. Vor dem Kauf gründlich durchchecken.

Abbildung 14.1: Thunder Chopper

Abbildung 14.2: Test drive

Abbildung 14.3: Airline

Abbildung 14.4: Block' n' Bubble

Abbildung 14.5: JO

Abbildung 14.6: Demon Stalker

Fortgeschrittenenteil

Kapitel 15

Ins Innere geschaut

Dem Computer auf der Spur. Im C 64 arbeitet eine bisher unbekannte Organisation. Sie operiert mit eigener Sprache in einer eigenen Stadt. Der geheimnisvolle Chef ist sehr pingelig und hat uns schon häufig genervt. Seine »Bande« besteht aus einer Menge Elektronik. Was tut sich im Innern des Computers? Seit einer Weile beschäftigt mich eine Frage: Wieso kann ich mit einer Tastatur Computerspiele ausprobieren und Mathe-Hausaufgaben lösen? Da muß etwas hinterstecken. Gehen wir auf die Jagd nach neuen Informationen und stecken unsere neugierigen Nasen hinein.

Der Einstieg in die Funktionsweise des C 64 verläuft über die Tastatur. Sie ist das Tor in die neue Welt, der Verbindungsmann zwischen Computer und uns. In Bild 15.1 sehen wir eine Zusammenfassung der Insassen und nahen Freunde des Computers. Die Tastatur ist nichts anderes als die Eingabe. Alle Eingaben werden in den Speicher getan und gleichzeitig auf der »Ausgabe«, dem Bildschirm, angezeigt. Bisher war die Sache recht einfach, aber wer organisiert das Ganze? Wer ist der Oberchef unseres Computers?

Computerhäuptling

Alle organisatorischen Aufgaben im Computer übernimmt die CPU: »Central Processing Unit«. Übersetzt heißt dieser Begriff: »Zentrale Steuereinheit«. Nichts passiert ohne ihr Wissen oder ihren Befehl, über allem hält sie ihre organisatorische Hand. Die kann ganz schön nerven! Die CPU ist das eigentliche »Gehirn« des Computers und ist ein sehr genauer und aufmerksamer Chef. Erinnert Ihr Euch an Fehlermeldungen wie »SYNTAX ERROR«, die uns so manch einen traurigen Moment beschert haben? Das hängt alles mit der CPU zusammen.

Gehen wir die Sache Stück für Stück durch. Wir machen eine Eingabe, zum Beispiel

```
C=20+10
```

Bild 15.1: Die Insassen des C 64 als Schema.

Zunächst geschieht nichts. Der Cursor bleibt hinter der 10 stehen und blinkt friedlich vor sich hin. Im Computer läuft folgendes ab: Die eingegebenen Zeichen werden im Speicher abgelegt und gleichzeitig wird eine Kopie dieser Zeichen auf dem Bildschirm ausgedruckt. Erst das Drücken der Return-Taste bringt sichtbare Bewegung in die Sache. Der Computer schreibt »READY« und setzt den Cursor an den Anfang der nächsten Zeile. Durch Drücken der Return-Taste wird der CPU die eingegebene Zeile mitgeteilt und dadurch ausgeführt. Wir haben die Wichtigkeit der Return-Taste besonders bei der Arbeit mit Programmen festgestellt. Erst sie teilt dem Computer die Veränderung oder Neueingabe einer Befehlszeile mit. Die eingegebene Zeile wird vom Speicher in die Leitstelle CPU übernommen und der entsprechende Befehl ausgeführt. In unserem Falle gibt die CPU den Befehl: Addiere 20+10 und lege den Wert unter dem Namen C im Speicher ab. Die CPU prüft die eingegebene Zeile und gibt alle weiteren Befehle. Der gespeicherte Wert ist jederzeit abrufbar. Die Befehlszeile

```
PRINT C
```

holt die Zahl aus den elektronischen Tiefen des Computers und druckt sie aus. Das Ergebnis ist »30«. Die CPU hat die Zeile »C=20+10« ausrechnen lassen und im Speicher abgelegt.

Kommen wir zu einer unangenehmen Eigenart der CPU. Sie erkennt im Verlauf eines Programms fehlerhafte Eingaben und druckt diese ohne Rücksicht auf den Gesundheitszustand des gefrusteten Programmierers aus. Stellt Euch vor: Drei Flaschen Cola und vier Tüten Chips wurden bei der Produktion eines langen Programms verbraucht. Nach RUN läuft alles glatt, doch plötzlich steigt der Computer (genauer die CPU) mit folgender Meldung aus dem Programm aus:

```
?SYNTAX ERROR IN 1580
```

Da haben wir sie, die nervende Genauigkeit der CPU. Irgendwo in Zeile 1580 haben wir eine für die zentrale Arbeitseinheit unverständliche Eingabe gemacht. Die Kontrolle ergibt: In Zeile 1580 steht

```
PWINT A
```

statt

```
PRINT A
```

PWINT ist ein unbekannter Befehl, das Programm »stürzt ab«, der Programmierer auch. Dieses ist die Hauptaufgabe der CPU. Sie kann Vergleichsoperationen, arithmetische Aufgaben und ähnliches ausführen. Leute, legt Euch bloß nicht mit ihr an, Ihr holt Euch heiße Ohren! Ich kann ein Lied davon singen. Die einzige Möglichkeit ist absolute Gründlichkeit.

Sehen wir uns Bild 15.1 genau an. Die durchgezogenen Linien stellen die eingegebenen und im Computer befindlichen Daten dar, gestrichelte Linien stehen für die Steuerbefehle der CPU. Die Eingabe C=20+10 wird im Speicher abgelegt. Gleichzeitig erhält die CPU eine Information über die neue Befehlszeile. Aufgrund dieser Nachricht erhält der Bildschirm die Anweisung, die eingegebenen Zeichen darzustellen. Das Ganze geht so schnell, daß die Buchstaben sofort auf dem Bildschirm erscheinen. Von den Arbeitsvorgängen im Computer bemerken wir nichts.

Namensgebung

Die Organisation des C 64 kennen wir jetzt. Wie sieht die Stadt aus, in der sie sich befindet? Freunde, ich kann Euch sagen, da gibt es eine Menge Geheimnisse. Wißt Ihr zum Beispiel, warum der C 64 C 64 heißt?

Auch wenn er nicht so aussieht: Der C 64 besteht aus 65536 einzelnen Bereichen, die vollgeschrieben werden können. Darum heißt er C 64. Klingt logisch, oder nicht? Ich behaupte »Ja«.

Im Computerbereich spricht man von Speicherkapazität. Genauso wie ein Schulheft zum Beispiel 80 leere Seiten hat, hat ein Computer leere Speicherkapazitäten. Ein solcher Speicher nennt sich »Arbeitsspeicher«. Er kann durch Programme vollgeschrieben werden. Je größer der Arbeitsspeicher eines Computers, desto komplizierter und größer können die Programme sein. Die Einheit, in der man den Speicher mißt, ist das »Byte«. Ein Byte kann mit einer Heftzeile verglichen werden: Der C 64 ist ein Heft mit 65536 Zeilen.

Ihr alle kennt den Begriff »Kilometer«. »Kilo« steht für 1000, ein Kilometer hat 1000 Meter, das weiß jeder. Den Speicherbereich eines Computers mißt man in »Kilo«-Byte, dieses Kilo hat mit dem Kilo von Kilometer nichts zu tun. Das Kilo des Speichers steht für 1024 und nicht für 1000. Das hat was mit binären Zahlen zu tun, ist aber jetzt nicht wichtig. Es wird mit einem »K« abgekürzt. Noch einmal: Der Speicherbereich eines Computers wird in Byte gemessen. 1024 Byte entsprechen einem Kbyte (sprich Ka-Byte). Der Arbeitsspeicher des C 64 besteht aus 64 Kbyte. Da geht ein Licht an: 64 Kbyte entsprechen 64*1024 Byte= 65536 Byte. Das ist des Rätsels Lösung. Der C 64 hat seinen Namen von der Größe seines Arbeitsspeichers.

An dieser Stelle tauchen zwei wichtige Basic-Befehle auf: PEEK und POKE. Mit diesen beiden Befehlen können wir die einzelnen Speicherstellen, die einzelnen Bytes des C 64, ansprechen. Der Befehl PEEK zeigt uns den Inhalt eines Bytes. Wenn wir ein Byte mit einer Heftzeile vergleichen, druckt PEEK den Inhalt der betreffenden Zeile aus. Mit POKE können wir den Inhalt verändern, neue »Worte« oder Befehle schreiben. Die 65536 Byte des C 64 haben alle eine Nummer, jede hat eine eigene Adresse, mit der wir uns immer zurechtfinden können. Stellt Euch vor, Ihr steht an einer langen Straße. Entlang dieser Straße gibt es in einer Linie 65536 durchnumerierte Häuser. Der Befehl PEEK öffnet die Haustür und Ihr könnt in das Innere des Hauses sehen, mit POKE könnt Ihr den Inhalt des Hauses verändern. Wenn Ihr den Befehl POKE eingebt, läuft eine Horde Möbelpacker in das Haus, schmeißt die alten Möbel raus und packt neue hinein. Das erste Speicher-Haus hat die Nummer 0, das letzte die Nummer 65535. (Wundert Euch nicht: Weil bei 0 zu zählen begonnen wird, hat das letzte Haus die Nummer 65535, insgesamt sind es 65536!)

Wanderwege im Computer

So viele Byte-Häuser, ein langer Weg! Jede Speicherstelle hat eine bestimmte Funktion, alles ist genauestens durchorganisiert. Nehmen wir zum Beispiel Byte 1024. Es gehört zum sogenannten »Bildschirmspeicher«. Der Bildschirm ist in viele kleine Abschnitte unterteilt, jeder Abschnitt kann einen Buchstaben aufnehmen. Die Adresse jedes Bildschirm-Abschnittes ist ein bestimmtes Byte. Die Position links oben in der Ecke des Bildschirms hat die Byte-Nummer 1024. Durch Ändern des Inhaltes von Byte 1024 können wir jeden beliebigen Buchstaben auf den Bildschirm rufen!

Paßt mal auf, gleich schlackert Ihr mit den Ohren. Für die Ohren-Gymnastik drücken wir zuerst [SHIFT] [CLR/HOME]. Dann gehen wir mit dem Cursor einige Zeilen nach unten. Folgenden Befehl eingeben:

```
POKE 1024,1
```

Links oben in der Ecke erscheint ein schnuckeliges »A« auf dem Bildschirm. (Achtung: Falls das A nicht sofort sichtbar wird, so müßt Ihr mit dem Cursor an die entspre-

chende Stelle fahren, dann seht Ihr es. Vom C 64 gibt es verschiedene Versionen. Die älteren Modelle zeigen die Zeichen erst durch Anfahren mit dem Cursor, das ist leider etwas umständlicher.) So einfach geht das. Wenn Ihr statt der 1 einen anderen Wert eingebt, entstehen beliebige Zeichen. Alle Zahlen von 0 bis 255 können verwendet werden. Probiert mal ein bißchen herum. Wenn Ihr wollt, setzt statt 1024 einen anderen Wert ein. Mit den Zahlen 1024 bis 2023 könnt Ihr jede Stelle des Bildschirms mit einem neuen Zeichen »beschreiben«. (Unter Bildschirm verstehen wir den eingerahmten dunkelblauen Teil, nicht den Rahmen drumherum.) Byte 1025 ist für den Abschnitt neben dem neuen »A« zuständig, 1026 rückt wieder einen Schritt nach rechts und so weiter (Bild 15.2).

Bild 15.2: Der Bildschirmspeicher teilt den Bildschirm in Zeichenplätze ein.

Die Position rechts unten in der Ecke hat die Nummer 2023. Folgendes Programm verdeutlicht das:

```
10 FOR A=1024 TO 2023
20 POKE A,81
30 NEXT A
40 GET B$:IF B$="" THEN 40
50 END
```

Es malt den Bildschirm mit Bällen voll. In Zeile 20 wird in einer Schleife ständig der Wert von A erhöht. Das bedeutet, wir füllen bei jedem Schleifendurchlauf eine andere Bildschirmposition mit dem Ball. Das Schöne an diesem POKE ist, daß wir nun gezielt Zeichen auf dem Bildschirm darstellen können. Probiert mal ein wenig herum.

Immer weiter hinein

Wir können auf ähnlich einfache Weise jedem Zeichen eine beliebige Farbe geben. Genauso wie es einen Bildschirmspeicher gibt, existiert ein Farbspeicher, der jedem Zeichen auf dem Bildschirm eine bestimmte Farbe zuweist. Der Farbspeicher beginnt bei 55296 (das »A« links oben in der Ecke) und endet bei 56295 (ganz rechts unten). Die Byte-Inhalte 0 bis 15 legen die Farben fest. Für ein rotes A links oben auf dem Bildschirm geben wir ein:

```
POKE 1024,1:POKE 55296,2
```

Das haut einen vom Hocker, stimmt's? Wie leicht der Umgang mit dem C 64 ist, wenn man nur einige Kleinigkeiten weiß. Durch Ändern der Werte in dieser Befehlszeile kann in jeder Bildschirmposition ein Zeichen von beliebiger Farbe gesetzt werden, ein nettes kleines Spiel. Hier die Liste der verschiedenen Farben:

```
0 Schwarz        6 Blau         12 Grau 2
1 Weiß           7 Gelb         13 Hellgrün
2 Rot            8 Orange       14 Hellblau
3 Türkis         9 Braun        15 Grau 3
4 Violett       10 Hellrot
5 Grün          11 Grau 1
```

Fügt unserem Ball-Programm noch ein paar Zeilen hinzu und Ihr werdet sehen, was machbar ist. Der Zufall hilft:

```
5 C=RND(-TI)
10 FOR A=1024 TO 2023
15 B=INT(RND(1)*14)
20 POKE A,81
25 POKE A+54272,B
30 NEXT
40 GET A$:IF A$="" THEN 40
50 END
```

Ein Start mit RUN bestätigt: Der C 64 malt den Bildschirm nun mit farbigen Bällen voll. Das Geheimnis liegt in Zeile 25. POKE A+54272 spricht immer die gleiche Bildschirmstelle an wie POKE A. POKE A ist jedoch verantwortlich für das Zeichen, welches auf dem Monitor ausgegeben werden soll. POKE A+54272 gibt dem Zeichen die

Farbe. Wenn Ihr mal nachrechnet, werdet Ihr sehen, daß in der ersten Schleife A+54272 gleich 55286 ist (1024+54272=55286). Durch diesen kleinen Trick sprechen wir also in jeder Schleife immer dieselbe Speicherstelle an. Die verschiedenen Farben werden per Zufall durch die Variable B erzeugt. Wenn Ihr alle Kapitel bis hierhin durchgearbeitet habt, solltet Ihr das Programm verstehen können. Habt Ihr jedoch einige Kapitel überflogen, kümmert Euch nicht dadrum. Hier geht es um die Speicherlandschaft des C 64 und nicht um Programmiertechniken. Nehmt Programme einfach mal so hin und laßt Euch faszinieren.

Die unterschiedlichen Speicher können einen verwirren, müssen aber nicht. Ein Byte bietet für die Daten des gewünschten Zeichens und dessen Farbe nicht genug Platz. Deshalb werden zwei Speicher benötigt. Der eine sagt: In Byte XY soll dieses Zeichen stehen. Die Farbe des Zeichens ist im Farbspeicher XYZ festgelegt. Die beiden Speicher liegen sehr weit auseinander, gehören aber eng zusammen. Dem Computer ist es völlig wurscht, in welchem der 65536 möglichen Speicher-Häuser die benötigten Daten liegen. Er muß nur wissen wo.

PEEK und POKE sind die Werkzeuge für die Arbeit mit dem Speicher. PEEK zeigt uns jederzeit den Inhalt eines Bytes. Im Unterschied zu POKE muß die Nummer des Bytes in Klammern gesetzt werden. Der Befehl

```
PRINT PEEK (1024)
```

zum Beispiel druckt die Zahl 1 aus, wenn links oben auf dem Bildschirm ein »A« steht, bei einem »B« erscheint 2. Steht dort kein Zeichen, also ein Leerzeichen, so gibt er 32 aus. Zu Beginn der PEEK-Zeile muß unbedingt PRINT stehen, sonst liest die CPU den Wert aus Byte 1024, teilt ihn uns aber nicht mit, klaro? In jedem Byte des Bildschirmspeichers befindet sich eine Zahl zwischen 0 und 255. Schauen wir in eine »leere« Bildschirmposition. Byte 1063 ist für den letzten Abschnitt der obersten Bildschirm-Zeile zuständig. Linsen wir mit

```
PRINT PEEK (1063)
```

in das Byte hinein. Es erscheint die Zahl 32. Das Byte enthält die Ziffer für ein Freizeichen: Der Wert 32 entspricht dem einmaligen Drücken der Space-Taste (bewirkt ein Freizeichen). Probieren wir das gleiche mit dem Farbspeicher aus. Das entsprechende Byte hat die Nummer 55335.

```
PRINT PEEK (55335)
```

Das Byte enthält die Zahl 14, was der Farbe Hellblau entspricht. Eine tolle Sache. Fassen wir zusammen: Der C 64 hat seinen Namen vom 64 Kbyte großen Speicher. Ein K entspricht der Zahl 1024, damit besitzt der C 64 65536 Byte. Die einzelnen Bytes können auf einfache Weise mit PEEK und POKE bearbeitet werden. Der gesamte Speicher ist genau eingeteilt. Alle Funktionen des Computers haben ihren eigenen Speicherbereich: wie zum Beispiel Bildschirm- und Farbspeicher. Der Bildschirm ist in

1000 einzelne Abschnitte unterteilt. Jeder Abschnitt kann einen Buchstaben aufnehmen. Den 1000 Bildschirm-Abschnitten entsprechen 1000 Byte im Speicher (Byte 1024 bis 2023), in denen das aktuell dargestellte Symbol in Form einer Zahl zwischen 0 und 255 gespeichert ist. Die Farbe jedes einzelnen Symbols ist im 1000 Byte großen Farbspeicher (Byte 55296 bis 56295) zu finden (er enthält eine Zahl zwischen 0 und 15).

An dieser Stelle stoßen wir auf eine wichtige Frage: Wie mache ich dem elektronischen Kasten vor mir eine Zahl klar? Es ist leicht gesagt: Mit einer 1 in Byte 1024 erhalte ich ein A links oben auf dem Bildschirm. Wie verarbeitet der Computer die Zahl 1?

Einfach zweifach!

Zunächst machen wir einen kleinen Test: Wir schalten den Computer aus und dann wieder an. Eins ist klar: Entweder der C 64 ist aus oder an, dazwischen gibt es nichts. Er benötigt Strom. Genau das gleiche Prinzip benutzt der C 64 für seine interne Datenverarbeitung. Mit Zahlen wie 255 kann der Rechner nichts anfangen. Er kann nur entscheiden, ob eine Sache an- oder ausgeschaltet ist. Alle Zahlen müssen dem Computer in Form von Informationen vermittelt werden, die entweder an oder aus sind. Das klingt alles ziemlich spanisch, oder?

Machen wir uns das Problem klar: Der Computer ist ein elektronisches Gerät, das zur Arbeit Strom benötigt. Wenn ich dem Computer eine Zahl übermitteln will, muß ich mir etwas einfallen lassen. Auch das wütendste Gebrüll meinerseits entlockt dem C 64 nur ein kleines Stirnrunzeln, er kapiert es nicht.

Es gibt eine Darstellungsart für Zahlen, die nur zwei Zeichen benötigt: Die Null und die Eins. Bevor wir diese Schreibweise kennenlernen, ist ein kleiner Ausflug in die Mathematik nötig. Es geht um »Potenzzahlen«. Unter einer Potenzzahl versteht der Mathematiker eine Zahl, die mit sich selber multipliziert wird. Ein Beispiel: Den Ausdruck 2*2*2 können wir mit Hilfe einer Potenzzahl darstellen. 2*2*2 bedeutet 2^3 (sprich 2 hoch 3). Die »Hochzahl« (die 3) ist die Zahl, die höher als die andere geschrieben wird (sagt ja schon der Name). Sie gibt an, wie oft die »Basiszahl« (die Zahl, die »unten« steht: in unserem Beispiel die 2) mit sich selber multipliziert werden soll. 2^4 bedeutet nach dieser Regel: Die Basiszahl 2 soll viermal mit sich selber multipliziert werden. 2*2*2*2. 2^1 bedeutet: Die Basiszahl 2 soll ein einziges Mal mit sich selber multipliziert werden: Das Ergebnis ist 2. Eine Ausnahme ist der Ausdruck 2^0. In diesem Falle ist das Ergebnis 1: 2^0=1. Das ist eine Festlegung, an der nicht zu rütteln ist: In der Mathematik ist jede Basiszahl hoch Null gleich 1: 500^0=1, 10^0=1, 1000000^0=1 und so weiter.

Ich weiß, dieser Abschnitt ist mit »Einfach zweifach« überschrieben und so einfach ist er nicht. Wartet mal ab, was wir gleich alles entdecken. Für uns ist nun die Hochzahlen-

Reihe von 2 besonders wichtig. Listen wir die Schreibweise und Ergebnisse der ersten acht Hochzahlen auf:

```
2^0 =   1
2^1 =   2
2^2 =   4
2^3 =   8
2^4 =  16
2^5 =  32
2^6 =  64
2^7 = 128
```

Mit diesen acht Zahlen können wir sämtliche Werte bis 255 darstellen. Ihr denkt jetzt vielleicht: das reicht! Erst erzählt der Typ was von Zahlen, die nur mit den Zeichen 0 und 1 dargestellt werden sollen und dann flippt er völlig aus: Er will alle Zahlen bis 255 als Potenzzahlen von 2 schreiben. Ich kann dieses Gefühl sehr gut verstehen, auch mein Gehirn schlägt einen Salto nach dem anderen. Trotz allem, nach diesem Kapitel verstehen wir den C 64 viel besser.

Paßt mal auf: Welche Zahl wollen wir ausprobieren? Fangen wir klein an. Die Zahl 5. Ich kann 5 in 4+1 zerlegen. Oben in der Liste sehe ich: 4 ist 2^2 und 1 ist 2^0. Da haben wir es: 5 = 4+1 = 2^2+2^1! Das gleiche Verfahren klappt mit allen Zahlen bis 255! Wie sieht es denn mit der Zahl 37 aus? 37 kann zerlegt werden: 37 = 32+4+1 = 2^5+2^2+2^1! Es klappt. In der Liste finden wir einige weitere Beispiele:

```
255 = 128+64+32+16+8+4+2+1
    = 2^7+2^6+2^5+2^4+2^3+2^2+2^1+2^0
 96 = 64+32  = 2^6+2^5
 25 = 16+8+1  = 2^4+2^3+2^1
```

Faszinierend! Probiert einige Zahlen selber aus.

Zimmerbelegung mit Einsen und Nullen

Haltet Euren Kopf fest, jetzt kommt der Hammer. Wir packen unser geballtes Wissen zusammen und erhalten die Datenverarbeitung des C 64. Vorhin stellten wir fest: In die Bytes des Bildschirm-Speichers passen Zahlen bis 255. Die Zahl kennen wir doch schon irgendwoher?! Richtig, Werte bis 255 können mit den ersten acht Potenzen der Zahl 2 dargestellt werden.

```
2^7+2^6+2^5+2^4+2^3+2^2+2^1+2^0 = 255
```

Was würdet Ihr sagen, wenn ein Byte acht »Zimmer« hat, in jedem »wohnt« eine Potenz von 2 (Bild 15.3)!

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
=	=	=	=	=	=	=	=
128	64	32	16	8	4	2	1

Bild 15.3: In jedem der acht Zimmer eines Bytes befindet sich eine Potenz von 2.

Wir erhalten alle benötigten Zahlen, indem wir die Inhalte der einzelnen Zimmer addieren. Die Zahl 25 ergibt sich aus der Addition der Zimmerinsassen 2^4, 2^3 und 2^1. Das können wir anders schreiben. Wir wissen, daß jede Zahl mit 0 multipliziert automatisch 0 ergibt. Deshalb können wir schreiben:

```
25 = 0*2^7+0*2^6+0*2^5+1*2^4
     +1*2^3+0*2^2+0*2^1+1*2^0
```

Diese Zeile ergibt

```
25 = 0     +0     +0     +1*2^4
     +1*2^3+0      +0     +1*2^0
```

Das Ganze tragen wir in die Zimmer unseres Bytes ein. Die Werte, die nicht benötigt werden, erhalten eine 0, die, die wir brauchen eine 1 (Bild 15.4). Da fällt es doch wie Schuppen von den Augen. Die Zahl 25 wird mit den Zeichen 0 und 1 dargestellt.

2^7 = 128	2^6 = 64	2^5 = 32	2^4 = 16	2^3 = 8	2^2 = 4	2^1 = 2	2^0 = 1	Bit
0	0	0	1	1	0	0	1	binär
			16	8			1	dezi.
			Ergebnis: 25					

Bild 15.4: Darstellung der Zahl 25 in einem Byte.

In den Byte-Zimmern, deren Bewohner wir für die Addition brauchen, steht eine 1. Alle anderen werden 0 gesetzt, weil wir sie nicht brauchen. Unsere Behauptung von oben stimmt. Wir können mit diesem Verfahren alle Zahlen bis 255 darstellen. Die Bilder 15.5 und 15.6 zeigen die beiden anderen Zahlen unserer Beispiel-Liste.

2^7 = 128	2^6 = 64	2^5 = 32	2^4 = 16	2^3 = 8	2^2 = 4	2^1 = 2	2^0 = 1	Bit
1	1	1	1	1	1	1	1	binär
128	64	32	16	8	4	2	1	dezi.
Ergebnis: 255								

Bild 15.5: 255 füllt jedes Zimmer mit einer 1.

2^7 = 128	2^6 = 64	2^5 = 32	2^4 = 16	2^3 = 8	2^2 = 4	2^1 = 2	2^0 = 1	Bit
0	1	1	0	0	0	0	0	binär
	64	32						dezi.
	Ergebnis: 96							

Bild 15.6: 96 als Byte-Darstellung.

Wenn wir die Zimmer weglassen, sieht die neue Schreibweise so aus. Beispiele:

```
255  11111111
96   01100000
25   00011001
105  01110001
24   00011000
87   01010111
0    00000000
```

Rechnet die Beispiele nach und probiert selber einige Zahlen aus. Die Sache ist ungewohnt, aber spannend.

Wir haben vieles gelernt: Jedes Speicher-Byte des Computers besteht aus acht Zimmern. Die Bewohner der Zimmer sind die Potenzen 0 bis 7 der Zahl 2. Bei der Eingabe einer Zahl in ein Byte wird in die benötigten Zimmer eine 1 geschrieben (das heißt: 1mal den Inhalt dieses Zimmers). In die anderen Zimmer kommt eine 0 (0mal den Inhalt dieses Zimmers). Jede Zahl läßt sich in Form einer Reihe von Einsen und Nullen darstellen. Diese Schreibweise nennt man »binäres Zahlensystem« oder »Dualsystem«. Sowohl »binär« als auch »dual« bedeuten »zwei« und weisen darauf hin, daß die Zahl 2 die Grundlage der neuen Darstellungsart ist.

Die Speicherzimmer eines Bytes werden in der Fachsprache »Bit« genannnt. Acht Bit faßt man als ein Byte zusammen, so wie wir acht Worte zu einer Zeile zusammenfassen würden. Bit ist die Abkürzung für »binary digit«, was auf deutsch »Binärzahl« heißt. Es ist die kleinste Darstellungseinheit im Binärsystem. Es kann nur zwei Werte aufnehmen, entweder 0 oder 1. Diese Bezeichnung ist verständlich, in ein Zimmer kommt entweder eine Eins oder eine Null. Ein Zimmer entspricht einer Binärzahl und wird deshalb Bit genannt. Ein Bit ist die kleinste Speichereinheit eines Computers. Das war ein bißchen viel auf einmal. Ganz deutlich kann man sich das machen, wenn man die Arbeitsweise des C 64 beleuchtet. Der C 64 arbeitet mit Strom. Er kann also nicht unterscheiden, ob etwas rot, grün oder blau ist, sondern nur, ob Strom vorhanden ist oder nicht. Befindet sich an bestimmten Stellen Strom, führt der C 64 eine Funktion aus. Mit binären Zahlen stellen wir im Prinzip nur da, wo Strom anliegt und wo nicht. Tatsächlich haben die Pioniere der Computer noch mit binären Zahlen programmiert. Damals wäre ein Computer wie der C 64 aber auch so groß gewesen wie ein Haus. Überprüfen wir unsere Kenntnisse an einem kleinen Beispiel.

Vorhin haben wir mit dem Befehl

```
POKE 1024,1
```

ein großes A auf dem Bildschirm plaziert. Mittlerweile ist uns klar: Der Computer wandelt die eingegebenen Zahlen in Dualzahlen um, die er nun im Speicher verarbeiten

kann. Die Eins wird in die Dualzahl 00000001 umgewandelt und in die acht Bit des Bytes Nummer 1024 getan. Das bedeutet: In jedes Byte-Zimmer (Bit) kommt eine Null, nur das kleinste erhält eine 1, weil 1*2^0=1 ist! Wenn wir ein »G« mit der Zahl 7 eingeben, so ändert sich die Bit-Belegung in 00000111 und so weiter. Der C 64 erkennt daran, welches Zeichen wir verlangen.

Wir haben einen riesigen Bogen geschlagen. Erinnert Ihr Euch an die Behauptung, der Computer könne nur Dinge verstehen, die an oder aus sind? Eine Null bedeutet »Aus«, eine Eins »An«.

Der C 64 schaltet die einzelnen Bits an oder aus und erkennt so die Zahl. Die Bit-Belegung der Zahl 7 00000111 bedeutet: Die ersten fünf Bits sind ausgeschaltet (ohne Strom), die Bits für 2^2, 2^1 und 2^0 sind eingeschaltet.

Jedes Bit hat einen eigenen Ein- und Ausschaltknopf, durch den der C 64 die gespeicherte Zahl genau festlegt.

Merkt Ihr, wie sich das Bild abrundet und die Fragezeichen kleiner werden? Ich weiß, es prasseln sehr viele Neuigkeiten auf uns ein, wir kommen kaum zum Luftholen und müssen manche Dinge zweimal lesen. Gleichzeitig wird das Computer-Wissen immer größer und bald ist der C 64 wie Knetmasse in unseren Händen!

Wer durch den Speicher will, muß verdammt lange laufen!

Das hätten wir geklärt. Ein Computer-Rundumschlag hat uns tiefen Einblick in die Datenverarbeitung des C 64 gewährt. Durch diese Aktion ist meine eigene Datenverarbeitung da oben durcheinandergeraten.

Sie will nicht mehr. Kein Pardon, bevor ich meine Bits zur Ruhe legen kann, muß ich sie mit einer Wanderung durch den Speicher strapazieren.

Wir kennen jetzt die Organistion und die Datenverarbeitung des C 64. Vom Speicher sind uns bisher 2000 Byte bekannt. Bildschirm und Farbspeicher haben genau festgelegte Speicher und Adressen.

Die Vermutung liegt nahe: Der gesamte Speicher ist nach einem genauen Muster geordnet. Schauen wir uns Bild 15.7 an. Es zeigt die Speicherlandschaft der 65536 Häuser auf einen Blick.

Bild 15.7: Die Speicheraufteilung des C 64 auf einen Blick.

Bevor wir richtig loswandern gibt es ein Problem. Wenn wir den Computer einschalten, fällt uns folgendes auf: Er meldet sich mit

```
   *** COMMODORE 64 BASIC V2 ***
 64K RAM SYSTEM 38911 BASIC BYTES FREE

READY
```

Diese Meldung erscheint bei jedem Einschalten. Das ist merkwürdig: Ein von uns gespeichertes Programm geht durch Ausschalten verloren. Die Bereitschafts-Meldung erscheint immer nach dem Einschalten. Das Geschreibsel auf dem Bildschirm kann nur von einem Programm erzeugt werden. Wieso geht dieses Programm nicht verloren, wenn der Computer ausgemacht wird?

Des Pudels Kern (aus Goethes Faust) ist: Es gibt zwei Arten von Speichern. Der eine wird durch Ausschalten gelöscht, der andere bleibt immer so bestehen, wie er ist. Er ist unangreifbar. Zwei neue Begriffe tauchen auf: ROM und RAM. Eins ist klar: ROM gehört zur »Speicherstadt« des Computers, hat aber nichts mit der italienischen Landeshauptstadt zu tun. ROM ist eine Abkürzung für »Read Only Memory« (auf deutsch: »Nur-Lese-Speicher«). Der Inhalt dieses Speichers ist fest im Computer »eingebaut« und wird deshalb durch Ausschalten nicht gelöscht. Wir können diesen Speicher nur lesen und ihn nicht verändern. Im ROM sind die Programme gespeichert, die für den Betrieb des Computers benötigt werden. Nehmen wir ein Beispiel: Irgendwo im Computer müssen Zeichen wie »A« und »Z« gespeichert sein. Das ganze Alphabet ist in der Form eines Programms im Computer enthalten. Es wäre schrecklich

umständlich, wenn wir nach dem Einschalten dem C 64 erst ein Programm eingeben müßten, das das Alphabet enthält. Aus diesem Grunde ist das Programm mit allen darstellbaren Zeichen im ROM gesichert. Es geht nie verloren und steht nach dem Einschalten sofort zur Verfügung. Das ROM beinhaltet auch das Programm, welches nach dem Anschalten des Computers die Bereitschaftsmeldung ausdruckt.

Auf der anderen Seite gibt es den sogenannten »RAM«-Speicher. Das Wort bedeutet »Random Access Memory« (deutsch: »Speicher mit beliebigem Zugriff«). Dazu gehören unter anderem auch Bildschirm- und Farbspeicher, denn sie können mit POKE nach Belieben verändert werden. In diesen Speicher werden unsere Basic-Programme geladen. Wenn wir den Computer ausschalten, verschwinden die Programme auf Nimmerwiedersehen.

Stellt Euch den Unterschied zwischen ROM und RAM so vor: Bei eingeschaltetem Computer ist im gesamten Speicher (ROM und RAM) Licht an, überall wird hektisch gearbeitet. Plötzlich wird der Computer ausgeschaltet: Im RAM legt sich alles sofort pennen und vergißt, was eben getan wurde. Alle Bewohner des RAM verlieren ihr Gedächtnis. Die Bewohner des ROM dagegen stört der ausgeschaltete Computer überhaupt nicht, sie haben nach dem Aufwachen ihr Gedächtnis noch.

Bild 15.7 ist unsere Landkarte. Auf ihr befinden sich alle wichtigen Informationen. Die kleinen Fähnchen mit den Zahlen geben die Nummern von Bytes an. Der erste Bereich geht von Byte 0 bis 1023. Diese Speicherstellen sind der »Notizblock« der CPU, man nennt diesen Notizblock »Zeropage«. Die Central Processing Unit legt hier wichtige Informationen ab. Der nächste markierte Bereich ist uns bereits bekannt: die Adressen des Bildschirm-Speichers. Kurz danach, ab Byte 2048, stoßen wir auf einen sehr wichtigen RAM-Speicher: den Basic-Speicher. In diesen Speicher kommen alle Basic-Programme. Wenn wir ein Basic-Programm schreiben, so wird es in den Bytes 2048 bis 40960 abgelegt. Erinnert Ihr Euch an die Einschaltmeldung? In der stand »38911 BASIC BYTES FREE«. Das entspricht genau dem Basic-Speicher und bedeutet: Dir stehen 38911 freie Bytes zum Speichern von Basic-Programmen zur Verfügung.

Doppelt und dreifach

Ab Speicherstelle 40960 stoßen die fröhlichen Wandersleut' auf eine Neuheit. Die Byte-Speicher-Häuser haben plötzlich mehrere Etagen. Bisher hatten sie nur ein Erdgeschoß, jetzt tauchen mehrgeschossige Häuser auf. Die Sache ist nicht weiter schlimm, der Computer weiß genau, was er wo findet, deshalb können die Häuser ruhig mehrere Stockwerke besitzen. Wenn er etwas aus dem ersten Stockwerk benötigt, geht er hoch und holt es sich. Kein Problem, gewußt wo!

Das Erdgeschoß ist weiterhin mit RAM besetzt, in den oberen Etagen machen sich verschiedene ROM-Speicher breit. Der einstöckige Speicherbereich 40960 bis 49152

enthält im ersten Stock den seit längerem bekannten Basic-Interpreter. Er ist nichts anderes als ein Computerprogramm, das die eingegebenen Basic-Befehle in Maschinensprache übersetzt. Ein besonderer Leckerbissen findet sich ab Byte 53248: ein zweistöckiger Speicher. Im ersten Stock finden wir alte Bekannte: zum Beispiel den Bildschirm-Farbspeicher und den vom Kapitel »Sound-Programmierung« bekannten Sound Interface Device (SID).

Der zweite Stock beherbergt das Zeichen-ROM: Alle auf dem Bildschirm darstellbaren Zeichen sind hier fest im Computer verankert. Von hier holt sich der C 64 die Informationen, wenn wir ein neues Zeichen in den Bildschirmspeicher POKEn. Alle Zeichen sind in Form der dazugehörigen Zahl abgelegt. Ihr erinnert Euch: Der Buchstabe »A« hatte die Ziffer »1«. Wenn wir in ein Byte des Bildschirmspeichers eine 1 eingeben, schaut der C 64 im Zeichen-ROM nach, welcher Buchstabe die Zahl 1 ist und schreibt ihn auf den Bildschirm.

Der letzte Speicher-Abschnitt geht von 57344 bis 65535. Er enthält ein Stockwerk über dem »Boden« einen besonders wichtigen ROM-Speicher: das Betriebssystem. Ohne das Betriebssystem ist der Computer eine nutzlose Anhäufung von Metall und Plastik, eine schreckliche Vorstellung. Von hier gehen die Steuerungsbefehle der CPU ab, denn das Programm des Betriebssystems ist nichts anderes als die CPU. Damit sind wir wieder am Anfang angekommen, beim pingeligen Chef des Computers. Von hier kommen die Fehlermeldungen »SYNTAX ERROR« und hier hat die Einschalt-Meldung ihren Wohnsitz. In diesem wichtigen ROM-Speicher sitzt das Gehirn des C 64. Es weiß über alle Vorgänge im Computer Bescheid, nichts entgeht ihm. Das Betriebssystem macht aus dem C 64 erst das, was er ist. Mit pingeligem Auge wacht der »C 64-Chef« über alle Vorgänge und fängt an zu mosern, wenn ihm etwas nicht gefällt.

Das kann mich überhaupt nicht schocken, denn nach diesem Kapitel ist meine Bit-Kapazität fast überschritten. Ich brauche unbedingt eine Pause. Meine letzte Tat besteht im Durchlesen des kleinen Lexikons der Speicherlandschaft und all ihrer Begriffe.

Das haben wir gelernt

Betriebssystem: Im Computer fest eingebautes Programm, das die inneren Abläufe und Datenverarbeitungen regelt. Das Betriebssystem ist das »Gehirn« des Computers.

Binäres Zahlensystem: Im binären Zahlensystem werden alle Zahlen durch Nullen und Einsen dargestellt. Hinter den beiden Ziffern verbergen sich die Potenzzahlen von 2. Im Computer bezeichnen die »Nullen« und »Einsen« eingeschaltete (1) und ausgeschaltete (0) Bits.

Bit: Abkürzung für »Binary Digit« (Binärzahl). Bezeichnet die kleinste Speichereinheit des Computers. Acht Bit ergeben ein Byte.

Byte: Zweitgrößte Speichereinheit des Computers, besteht aus acht Bit. Ein Byte kann Werte zwischen 0 und 255 annehmen, die in Form von Dualzahlen gespeichert werden.

Kbyte: (sprich Ka-Byte) »K« ist die Abkürzung für 1024. Ein Kbyte entspricht 1024 Byte.

PEEK: Basic-Befehl, mit dem der Inhalt eines Bytes abgefragt werden kann.

POKE: Basic-Befehl, mit dem Speicherstellen (Byte) neue Inhalte zugeordnet werden können.

RAM: Abkürzung für »Random Access Memory« (Speicher mit beliebigem Zugriff). Der RAM-Speicher steht dem Anwender frei zur Verfügung und kann beliebige Inhalte erhalten.

ROM: Abkürzung für »Read Only Memory« (Nur-Lese-Speicher). Der ROM-Speicher kann vom Anwender nicht verändert werden, er wird nur gelesen. Enthält fest im Speicher verankerte Programme (Betriebssystem, Basic-Interpreter...), die auch nach dem Ausschalten des Computers nicht verlorengehen.

Kapitel 16

Abfallbeseitigung auf Disketten

Meine Disketten nerven mich.

Auf einigen befinden sich unendliche Mengen von Programm-Ruinen, die ich nicht benötige. Der Speicherplatz für brauchbare Programme wird knapp. Ich muß mir etwas einfallen lassen.

Auf der Suche nach Müllmännern und Putzmöglichkeiten stoße ich auf sehr nützliche Befehle. Die Disketten-Station erweist sich als schlummernder Riese.

Ihr kennt das bestimmt. Beim Programmieren entstehen häufig Listings, die später verbessert werden. Diese überflüssigen Programme gammeln auf der Diskette herum und keiner weiß etwas mit ihnen anzufangen.

Im Anschluß an unser Eiskauf-Programm habe ich ein Programm namens »Autokosten« geschrieben. Das fertige Programm (»Autokosten 5«) hat vier unnütze, nicht funktionsfähige Brüder. Zunächst stellte ich fest: Ich kann mit dem normalen SAVE-Befehl kein Programm unter demselben Namen zweimal speichern.

Das erste Programm »Autokosten« ließ keine weitere Speicherung unter diesem Namen zu. Deshalb habe ich jede neue Version mit einer Nummer versehen, bis die Nummer 5 endlich problemlos funktionierte.

Das Ergebnis ist: Ich habe vier nervende, Speicherplatz-klauende Programme auf meiner Diskette, die ich nicht mehr haben will.

Bild 16.1: Directory einer Diskette.

Programm-Mülleimer

Bild 16.1 zeigt uns das Inhaltsverzeichnis meiner Diskette: Ein buntes Durcheinander von numerierten Programmen. Die überflüssigen Programme »Autokosten« will ich loswerden. Auf der Suche nach Mülleimern bin ich auf folgenden nützlichen Befehl gestoßen:

```
OPEN 15,geräte#,15,"S0:dateiname"
:CLOSE 15
```

Da geht einem der Hut hoch. Wieso müssen sich alle so kompliziert ausdrücken? Der Befehl

```
OPEN 15,8,15,"S:AUTOKOSTEN 1":CLOSE 15
```

ist viel einfacher. Die Disketten-Station legt los und meldet sich mit READY zurück. Ich kontrolliere das Ergebnis mit

```
LOAD"$",8
```

und

```
LIST
```

Ah, Zufriedenheit kreist um mein Haupt: Das ungeliebte Programm »Autokosten 1« ist aus dem Directory der Diskette verschwunden. Auf dem gleichen Wege werfe ich die restlichen Programme in die ewigen Speichergründe. Das klappt ja prima! Die ersten drei Zahlen im neuen Befehl sind von der Disketten-Formatierung bekannt. Der Befehl CLOSE schließt den Befehlskanal 15 und muß unbedingt eingegeben werden, da Kanal 15 sonst offen bleibt. Die größte Neuigkeit finden wir innerhalb der Gänsefüßchen »S:AUTOKOSTEN 1".

An die Stelle von »Autokosten 1« können wir den Namen jedes Programms setzen, das wir löschen wollen. Das große »S« vor dem Doppelpunkt ist ein Kürzel für das Wort »Scratch« und bedeutet »ausradieren, löschen«. Scratch erlaubt uns das Löschen jedes beliebigen Programms.

Meine Diskette ist deutlich aufgeräumter als vorher, trotzdem ärgert mich etwas. Ich benötige keine Ziffer hinter dem Autokosten-Programm, da es jetzt das einzige auf der Diskette ist! Ich will den Namen des Programms abändern, von »Autokosten 5« zu »Autokosten«.

Es gibt einen Befehl, der die neue Namensgebung möglich macht. Die Disketten-Station erweist sich als ein leistungsfähiges Kerlchen. Der gesuchte Befehl lautet:

```
OPEN 15,8,15,"R:neuer Name=
                alter Name":CLOSE 15
```

Praktischerweise bleibt ein Teil des Befehls bestehen, der Ausdruck in Anführungsstrichen hat sich geändert. Das »R« steht für »RENAME«, was auf deutsch »Umbenennen« heißt. Die Reihenfolge der verschiedenen Namen müssen wir genau beachten, sonst veräppelt uns die Disketten-Station. Zuerst wird der neue Name eingegeben, dahinter kommt der »alte«. Mein Ziel ist klar: Der neue Name ist »Autokosten«, der alte »Autokosten 5«. Ich tippe ein:

```
OPEN 15,8,15,"R:AUTOKOSTEN=
                AUTOKOSTEN 5":CLOSE 15
```

Wieder überprüfe ich den Vorgang durch Aufrufen des neuen Directorys. Es funktioniert. Nur meine schlappen Gehirnzellen können sich die Reihenfolge »neuer Name=alter Name« nicht merken und ich vertausche beide ständig. Nach einer Weile habe ich diese Klippe umschifft. Es kann weitergehen.

Ein schwieriger Klammeraffe

Ich bin neugierig geworden. Von diesen Befehlen hatte ich bisher nie etwas gehört. Mal schauen, was die Disketten-Station alles zu bieten hat. Habt Ihr schon einmal etwas vom »Klammeraffen« gehört? Es gibt einen Befehl, mit dem verbesserte Programme unter dem gleichen Namen gespeichert werden können. Mit diesem Befehl wäre es gar nicht zu der Ansammlung von Autokosten-Programmen gekommen, da die neue Version von der alten »überspeichert« wird. Der Befehl lautet:

```
SAVE"@:Programm",8
```

Der Klammeraffe ist eine Taste, die sich neben der »P«-Taste befindet. Sie trägt ein Zeichen, das wie ein kleines »a« aussieht, um das man eine Schleife gezogen hat. Aber Vorsicht: Ich habe mich für Euch bei Profis erkundigt. Sie haben mir mit Tränen in den Augen vom Gebrauch dieses Befehls abgeraten: Er birgt die Gefahr, sämtliche Daten des betreffenden Programms zu verlieren. Das würde bedeuten: Alles neu programmieren. Wir gehen lieber den oben beschriebenen Weg mit den durchnumerierten Programmen, als einmal die gesamte Arbeit zu verlieren. Der Klammeraffen-Befehl arbeitet manchmal ungenau: Ihr besitzt nach einem »abgestürzten« Klammeraffen-SAVE weder die alte Version des Programms noch die neue. Bei kleinen Programmen ist das Risiko tragbar, bei langen Listings gehen wir lieber den umständlicheren Weg!

Viele Programme sollten zur Sicherheit doppelt vorhanden sein. Die Disketten-Station bietet aus diesem Grunde einen Kopier-Befehl an.

```
OPEN 15,8,15,"C0:neue Datei=
               0:alte Datei":CLOSE 15
```

Der Befehl sieht wieder nur innerhalb der Anführungszeichen anders aus. Nach dem C und nach dem Gleichheitszeichen muß eine Null eingegeben werden. Mit dem neuen Befehl wird das entsprechende Programm geladen und unter dem angegebenen Namen wieder auf die Diskette gespeichert. Auf diese Weise erhalten wir eine Sicherheitskopie (englisch »Backup«) des wichtigen Programms. Hier ein Beispiel: Ich will mein »Autokosten«-Programm unter dem Namen »Wagenkosten« sichern. Eingabe von

```
OPEN 15,8,15,"C0:WAGENKOSTEN=
               0:AUTOKOSTEN":CLOSE 15
```

genügt. Ihr müßt die »0« direkt hinter das Gleichheitszeichen setzen, sonst funktioniert die Sache nicht richtig.

Von Fragezeichen und Sternchen

Da strahlen mich neue Möglichkeiten an, ich kann alle Disketten aufräumen und endlich Ordnung schaffen. Trotz allem, vom LIST- und LOAD-Tippen habe ich fast Blasen

an den Fingern. Muß ich immer die ganzen Befehle eintippen oder gibt es Hilfsmittel für Arbeit und Umgang mit bestimmten Programmen? Wir als wahre Computer-Faulpelze lernen jede nur erdenkliche Programmier-Hilfe kennen.

Ein kleines und praktisches Hilfsmittel ist das Sternchen »*«. Es kürzt lange Programmnamen ab. Nehmen wir zum Beispiel unser Programm »Wagenkosten«, ein schrecklich langer Name. Bisher haben wir Programme immer mit vollem Namen geladen. Das Sternchen erlaubt folgenden Befehl:

```
LOAD"WAGEN*",8
```

Die Disketten-Station sucht nach einem Programm, das mit »Wagen« beginnt. Vorsicht, aus Faulheit wird schnell Mehr-Arbeit! Das Laufwerk lädt das erste Programm, das auf die Suchbeschreibung paßt. Dabei ist es völlig egal, ob das Programm »Wagenrennen« oder »Wagenkosten« heißt. Faule Leute müssen hellwach sein, sonst gehen sie baden.

Wo wir gerade bei Arbeitsersparnis sind: Basic-Befehle lassen sich auch abkürzen. Es muß nicht immer LIST oder LOAD eingegeben werden. LOAD und LIST können wir auf zwei Buchstaben verkürzen. Bei LOAD geben wir erst das »L« ein, dann drücken wir gleichzeitig die Tasten [O] und [SHIFT]. Auf dem Bildschirm erscheint anstelle der »O« ein grafisches Zeichen. Diese zwei Zeichen haben die gleiche Funktion wie LOAD. Probiert das mal aus. Es funktioniert auch bei LIST: Statt LIST geben wir »L« und dann [SHIFT] [I] ein. Kein Problem. Einzelne Befehle können durch Verwendung der SHIFT-Taste abgekürzt werden. Nach dem Anfangsbuchstaben wird anstelle des zweiten Buchstabens die Kombination <zweiter Buchstabe+SHIFT> gedrückt. Nie wieder wunde Finger.

Hier die Lösung für ein weiteres Problem: das Laden bestimmter Teile des Directorys. Beim Programmieren ist es oftmals wichtig zu wissen, wie viele Programme einer bestimmten Art auf der Diskette enthalten sind. Wir müssen nicht das ganze Directory laden, wenn wir die Zahl der Autokosten-Programme auf der Diskette wissen wollen. Mit

```
LOAD"$:PROGRAMM*",8
```

werden alle Wünsche erfüllt. Für »Programm« wird das gesuchte Programm(-Feld) eingegeben. Nehmen wir an, auf unserer Diskette befinden sich mehrere »Autokosten«-Programme. Die genaue Anzahl haben wir vergessen. Der Befehl

```
LOAD"$:AUTOKOSTEN*",8
```

ruft alle unter dieser Bedingung gespeicherten Programme auf den Bildschirm: Ob es ein einziges Programm ist, oder die Reihe bis »Autokosten 36« geht, spielt keine Rolle.

Es geht auch ohne!

Ich staune immer wieder. Beim Kauf meiner Disketten-Station 1571 habe ich mich über den hohen Preis geärgert. Mit der Zeit freundete ich mich mit ihr an, weil sie schnell, einfach und zuverlässig arbeitet. Jetzt merke ich, daß ich mit ihr auch einen kleinen Computer gekauft habe.

Die Disketten-Station kann zum Beispiel Daten ohne ein langes Computer-Programm speichern und wieder auslesen. In Bild 16.1 gibt es kleine Auffälligkeiten. Ein Punkt aus meiner Directory-Liste unterscheidet sich von den anderen: »SEQ-Datei«. Hinter allen anderen Listenpunkten steht »PRG«, nur hinter »SEQ-Datei« steht »SEQ«. Was hat das zu bedeuten?

Die Bedeutung von »PRG« ist klar: Es ist eine Abkürzung für »Programm«. Die mit einem PRG gespeicherten Dateien enthalten Computer-Programme. Sie werden als Ganzes geladen oder gespeichert, dazwischen gibt es nichts. »SEQ« ist eine Abkürzung, die zu völlig neuen Fähigkeiten des Disketten-Laufwerks führt: Unter dem Begriff »sequentielle Datei« können Daten direkt auf Diskette gespeichert werden, die Diskette wird zu einem leicht benutzbaren »Notizblock«.

In einer sequentiellen Datei werden Daten hintereinander, wie an einer langen Schnur, auf Diskette geschrieben. Auf dem Datenträger entsteht eine lange »Wurst« von Informationen. Diese »Wurst« kann jederzeit verlängert werden. Klingt nach Bahnhof, was? Ein kleines Beispiel wirkt Wunder.

Der Umgang mit sequentiellen Dateien erfordert einige Grundlagen. Die bisher kennengelernten Disketten-Befehle begannen alle mit

```
OPEN 15,8,15
```

Die Zahlen sehen wir uns genauer an. Die erste Ziffer muß eine Zahl zwischen 1 und 127 sein. Sie ist eine Dateinummer, die wir frei wählen können. Die Acht ist die uns bereits bekannte Adresse der Disketten-Station (erinnert Euch an Befehle wie SAVE "EISMANN",8). Die letzte Zahl steht für einen bestimmten Kanal. An dieser Stelle können Zahlen von 0 bis 15 stehen. Die Kanäle 0 und 1 sind für LOAD und SAVE reserviert. Nummer 15 ist der Kommandokanal, der zum Beispiel bei der Disketten-Formatierung verwendet werden muß. Wir können zwischen Zahlen von 2 bis 14 frei wählen. Diese Daten müssen nicht immer im Kopf behalten werden. Es reicht, wenn wir uns folgendes merken: Die erste Zahl kann zwischen 1 und 127 liegen, die zweite muß 8 sein und die dritte wird aus dem Bereich 2 bis 14 gewählt. So, jetzt legen wir los. Wir tippen ein

```
OPEN 1,8,2,"EINTRAG,S,W"
```

Die Disketten-Station sirrt los. Achtung: Die brennende Diode auf dem Disketten-Laufwerk zeigt an: »Es können Daten eingetragen werden.« Normalerweise geht die

Diode wieder aus, jetzt bleibt sie an. Wir haben den »Notizblock« Diskette geöffnet. Mit

```
PRINT#1,"HENNINGS ERSTER EINTRAG"
```

schreibe ich meine erste Information auf die Diskette. Es tut sich überhaupt nichts! Habe ich etwas falsch gemacht? Nein, der Datentransport läuft geräuschlos ab. Überprüfen wir die Eintragung mit folgendem Programm. Achtung: Vorher muß

```
CLOSE 1
```

eingegeben werden, um den Kanal zu schließen. Dieses kleine Programm ruft meine Eintragung auf den Bildschirm

```
10 OPEN 1,8,2,"EINTRAG,S,R"
20 INPUT#1,A$
30 PRINT A$
40 CLOSE 1
RUN
```

Auf dem Bildschirm erscheint

```
HENNINGS ERSTER EINTRAG
```

Ich bin total baff, wo kommt das denn auf einmal her? Nachdem ich meine Überraschung überwunden habe, stellen sich zwei Fragen:

1. Was bedeutet das ganze Geschreibsel da oben?

2. Es ist ja interessant, aber welchen praktischen Nutzen habe ich davon?

Dem Geschreibsel rücken wir gleich auf den Pelz. Der praktische Nutzen ist folgender: Stellt Euch vor, wir schreiben ein Programm, das von Daten abhängt, die sich ändern. Ein solches Programm kann zum Beispiel eine Adreßdatei sein, mit der Ihr sämtliche Adressen Eurer Freunde verwaltet, oder ein Archiv für Eure Schallplatten-Sammlung.

Im Normalfall müßt Ihr bei einer neuen Adresse das halbe Programm umschreiben, Ihr müßt für die Eingabe der neuen Adresse in das Programm eingreifen. Wie häufig werden neue Adressen benötigt. Eine sequentielle Datei erspart eine Menge Arbeit.

Die Adreßverwaltung liest einfach die in der sequentiellen Datei gespeicherten Namen und Ihr könnt nach Bedarf Adressen an das Ende der Wurst hängen. Ihr greift dabei nicht in das Programm ein, sondern verlängert nur die sequentielle Datei, indem Ihr die neue Adresse in die Datei eintragt.

Datei-Schlüssel

So, jetzt zu den einzelnen Befehlen. Der erste war

```
OPEN 1,8,2,"EINTRAG,S,W"
```

Durch diese Zeile öffnet die Disketten-Station (8) im frei wählbaren Kanal 2 eine Datei mit der Nummer 1. Sie erhält den Namen »EINTRAG«. Die beiden Buchstaben am Ende des Befehls sind Abkürzungen. »S« bedeutet »sequentielle Datei«. Diese Information ist für die Verarbeitung der eingehenden Daten wichtig. Das »W« steht für »Write« (auf deutsch »Schreiben«) und gibt an: Es werden Daten in die Datei eingetragen. Der Befehl zum Eingeben ist

```
PRINT#1,"DATEN"
```

Dieser Befehl ist dem herkömmlichen PRINT sehr ähnlich. Während das normale PRINT Zeichen auf den Bildschirm ausgibt, druckt PRINT# auf ein anderes, von uns definiertes Gerät. Das Zeichen »#« ist die amerikanische Abkürzung für »number«, zu deutsch also »Nummer«. Daher lesen wir PRINT# »PRINT NUMMER« oder »PRINT NUMBER«. PRINT#4,"HENNING" würde zum Beispiel meinen Namen auf einem Drucker ausdrucken lassen. Demnach gibt die Zahl hinter dem Nummernzeichen »#« an, auf welches Peripherie-Gerät geschrieben werden soll. Da der PRINT#-Befehl zwischen einer OPEN- und CLOSE-Anweisung steht, erkennt der C 64, daß nicht ein Gerät angesprochen werden soll, sondern eine Datei mit der Nummer 1. Hinter dem Befehl geben wir nach einem Komma den Eintrag in Anführungszeichen ein, das ist alles. Jede weitere Eingabe verläuft nach dem gleichen Muster.

Eine Sache ist besonders wichtig: Die von uns geöffnete Datei muß am Ende wieder geschlossen werden. Sonst laufen wir Gefahr, die gesamte Datei zu verlieren. Eine noch nicht geschlossene Datei wird im Directory mit einem »*« gekennzeichnet. Die Daten sind verloren. Dateien immer schließen!

```
CLOSE 1
```

schließt jeden Arbeitsvorgang ab.

Das Auslesen der gespeicherten Daten benötigt ein kleines Programm, das uns anfangs etwas spanisch vorkommt. Machen wir uns die einzelnen Befehle klar.

```
10 OPEN 1,8,2,"EINTRAG,S,R"
20 INPUT#1,A$
30 PRINT A$
40 CLOSE 1
```

Zeile 10 ist bis auf das »R« bekannt. Es steht für »Read« und bedeutet »Lesen«. Klar, eben haben wir geschrieben, jetzt wird gelesen. In Zeile 20 liest der Computer mit INPUT#1 den Inhalt von Datei Nummer 1 und legt ihn unter der Bezeichnung A$ in

seinem Speicher ab. Zeile 30 druckt diesen Speicher-Inhalt aus und Zeile 40 schließt die Datei wieder.

Tja, Computer gehören einem merkwürdigen Völkchen an. Da arbeitet und spielt man lange Zeit mit dem Kasten herum, plötzlich zeigt er einem völlig unbekannte Möglichkeiten. Diese Form der Disketten-Benutzung ist mir fremd. Ich fasse mein neues Wissen zusammen: Disketten können ohne lange Programme mit Daten beschrieben werden. Eine Möglichkeit sind »sequentielle Dateien«, die alle Daten in Form einer langen Schnur auf der Diskette ablegen. Diese Technik findet in Programmen Anwendung, die mit wechselnden Eintragungen arbeiten, zum Beispiel Adreßdateien. Der Computer liest auf einen kleinen Befehl hin die gesamte Reihe der Informationen ab und arbeitet mit ihnen. Wir können durch PRINT# nach einer OPEN-Anweisung unabhängig vom Programm neue Daten eingeben. Wenn Daten gelesen werden sollen, muß dies in der Befehlszeile durch ein »R« für »Read« angegeben werden. Bei der ersten Eintragung benutzen wir den Buchstaben »W« für »Write« gleich »Schreiben«.

Genug der Theorie. Eine Kleinigkeit müssen wir noch lernen, dann wenden wir uns einer tollen Sache zu: einem Kopierprogramm für Disketten. Wie können wir mehrere gespeicherte Daten von der Diskette lesen? Zunächst geben wir in unsere sequentielle Datei zwei neue Datensätze ein.

```
OPEN 1,8,2,"EINTRAG,S,A"
PRINT#1,"HENNINGS ZWEITER EINTRAG"
PRINT#1,"HENNINGS DRITTER EINTRAG"
CLOSE 1
```

Hier müssen wir aufpassen. Der Buchstabe »W« wird nur bei der ersten Eintragung in eine Datei verwendet. Tragen wir später etwas ein, wird anstelle des »W« ein »A« (merkt es Euch mit »Anfügen«) eingegeben.

Insgesamt befinden sich in der Datei drei Einträge. Das Auslesen der Daten geschieht mit dem bekannten Programm. Es muß an die veränderte Zahl angeglichen werden: Statt eines Eintrags sollen drei ausgelesen werden. Die nötige Erweiterung ist eine FOR-NEXT-Schleife in Zeile 15 und 35:

```
10 OPEN 1,8,2,"EINTRAG,S,R"
15 FOR I=1 TO 3
20 INPUT#1,A$
30 PRINT A$
35 NEXT I
40 CLOSE 1
RUN
```

Der C 64 fährt die Schleife dreimal ab und liest durch FOR...NEXT der Reihe nach die drei Eingaben aus. Auf dem Bildschirm erscheint in einer irren Geschwindigkeit

```
HENNINGS ERSTER EINTRAG
HENNINGS ZWEITER EINTRAG
HENNINGS DRITTER EINTRAG
```

Das klappt ja vorzüglich. So, Leute, genug gelernt. Erst die Arbeit, dann das Vergnügen. Jetzt gibt es einen Leckerbissen für alle Freunde des schnellen und sauberen Kopierens. Ein Kopierprogramm der Extra-Klasse.

Kopieren mit Komfort

»Wozu ein Kopierprogramm?« werdet Ihr fragen, wo wir doch alles selbst können mit den gerade gelernten Befehlen. Ein solches Programm hilft aber sehr, Disketten noch einfacher zu organisieren. Stellt Euch vor, Ihr wollt die Diskette eines Freundes vollständig kopieren. Mehrmals müßte ein und derselbe Kopiervorgang wiederholt werden, bis alle Programme kopiert worden sind. Das Programm Super-Copy, das auf der Diskette zu diesem Buch vorliegt, nimmt Euch solch stupide Arbeit ab. Es wurde im 64'er Sonderheft 26 »Rund um den C 64« veröffentlicht.

Nach dem Laden mit:

```
LOAD"SUPER-COPY",8
```

wird es einfach mit RUN gestartet. Es erscheint ein Menü mit den Punkten:

```
1. DIRECTORY          4. SCRATCHEN
2. KOPIEREN           5. VALIDIEREN
3. FORMATIEREN        6. ENDE
```

Wir wählen einzelne Funktionen durch Drücken der entsprechenden Ziffer-Tasten. Der Punkt »DIRECTORY« ist klar, das Inhaltsverzeichnis der Diskette wird angezeigt.

Eine entscheidende Erleichterung für uns ist der Punkt 2: KOPIEREN. Nach Drücken von [2] fordert der C 64 uns auf, die Quelldiskette einzulegen, also die Diskette, auf der sich die Programme befinden, die wir kopieren wollen. Nach Druck auf irgendeine Taste erscheinen nun alle Programme dieser Diskette nacheinander auf dem Bildschirm. Mit [J] oder [N] können wir nun bestimmen, welche Programme kopiert werden sollen. Nachdem wir das getan haben, erscheint folgendes Menü:

```
1. DIRECTORY
2. FORMATIEREN
3. VALIDIEREN

***SPACE***
FÜR WEITER
```

Nun können wir in Seelenruhe eine Zieldiskette aussuchen, eine Diskette also, auf die wir kopieren wollen. Wir können uns das Directory der Zieldiskette ansehen oder auch neu formatieren. Auf jeden Fall landen wir wieder bei diesem Menü. Erst nach dem Druck auf die Leertaste geht es weiter. Der C 64 meldet sich mit:

```
READING PROGRAMMNAME
```

während die Disketten-Station das Programm (PROGRAMMNAME) einlädt. Nun haben wir zwei Möglichkeiten, wir können alle Programme (Files) ohne Pause nacheinander einladen oder aber den Kopiervorgang in Einerschritten vollziehen, das bedeutet, jeder Kopiervorgang muß neu bestätigt werden. Dann geht es endlich los. Fehler werden rechtzeitig angegeben. Sind alle Files kopiert, erscheint die Meldung »KOPIE FERTIG«.

Bei 3., FORMATIEREN, müssen wir, wie schon bekannt, den Namen der Diskette und, mit einem Komma getrennt, die Identitätsnummer (ID) eingeben. Die Diskette wird dann direkt formatiert. Deshalb gebt acht, ob nicht zufällig die Quelldiskette in der Floppy-Station liegt.

Die Funktion 4., SCRATCHEN, ist dem Punkt »Kopieren« ähnlich, nur daß hier nicht kopiert, sondern gelöscht wird. Nacheinander werden alle Programme aufgelistet und wir entscheiden mit [J] oder [N], ob ein File gelöscht werden soll. So ein Mist, da habe ich doch ein falsches Programm markiert, nun ist es zu spät. Falsch! Bevor »Super-Copy« loslegt, fragt es uns sicherheitshalber nochmal und wir können unsere Eingabe verändern.

5., VALIDIEREN, checkt kurz ab, ob auch alles ordnungsgemäß vonstatten ging.

So, bevor Ihr nun in diesem Buch weiter Wissen ansammelt, kopiert erst mal kräftig von Freunden. Doch Vorsicht: Das selbstgeschriebene Programm von Eurem Freund Egon könnt Ihr mit seiner Erlaubnis kopieren, gekaufte, kommerzielle Programme nicht. Außerdem könnt Ihr mit »Super-Copy« höchstens 32 Programme auf einmal kopieren, die eine Gesamtlänge von maximal 234 Blöcken nicht überschreiten dürfen. Bis zum nächsten Kapitel dann (einfach umblättern)!

Das haben wir gelernt

Scratchen: Heißt Löschen. Mit dem Befehl

```
OPEN 15,8,15,"S:PROGRAMMNAME":CLOSE 15
```

können wir Programme von einer Diskette löschen.

Rename: Erlaubt das Umbenennen von Programmen auf der Diskette:

```
OPEN 15,8,15,"R:NEU=ALT":CLOSE 15
```

nennt das Programm mit dem Namen »ALT« in »NEU« um.

***:** Kürzt einen Programmnamen ab.

```
LOAD"AUTO*",8
```

lädt das erste Programm, dessen Name mit »AUTO« beginnt.

Sequentielle Datei: Mit einfachen Befehlen können wir eine sequentielle Datei aufbauen. Daten werden dabei hintereinander auf die Diskette geschrieben.

Kapitel 17

Der C 64 und seine Umwelt - Messen, Steuern, Regeln

Dem C 64 geht ein Licht an. Wir lernen die Theorie für den Anschluß des C 64 an eine Modelleisenbahn und damit an seine Umwelt. Nun müßt Ihr nicht gleich Panik kriegen. Alles ist ganz einfach und auch für uns Laien verständlich. Der C 64 zeigt weitere Stärken: Messen – Steuern – Regeln. In der Stadt Hameln zum Beispiel gibt es den Verein für angewandten Umweltschutz, der mit Hilfe des C 64 radioaktive Strahlung mißt und auswertet. Viele solche Messungen lassen sich mit anderen, sogar wesentlich teureren Computern, nur schwer machen. Damit auch wir ein Wörtchen mitreden können, werden wir uns das mal genauer ansehen.

Der C 64 bietet über verschiedene Anschlüsse Verbindung zur Außenwelt. In der Computerwelt nennt man diese Anschlüsse »Ports«. Der Joystick-Port ist also nichts anderes als ein Joystick-Anschluß. Doch der C 64 hat außer den beiden Joystick-Anschlüssen noch andere Ports, deren Funktionen wir uns gleich ansehen. Nebenbei erlernen wir die Programmierung des User-Ports. Eine neue Welt tut sich auf: die Lichtorgel des großen Bruders, die Modelleisenbahn oder Mutters Waschmaschine, alles messen, steuern und regeln wir mit dem C 64.

Bevor richtig programmiert wird, brauchen wir wie immer Grundlagen. Eine kleine »Tour-de-C 64« zeigt die neuen Möglichkeiten. Düsen wir los.

Er kam, sah und staunte

Auf der rechten Seite des faszinierenden Plastik-Kumpels finden wir einige altbekannte Anschlüsse: zwei Steckplätze für Joysticks, den Power-Knopf und die Stromversorgung. Das ist ein alter Hut. Interessanter wird es auf der Rückseite. Dreht den C 64 mit der Tastatur zur Wand, so habt Ihr freien Blick auf die Anschlußmöglichkeiten der Rückseite. Ganz links entdecken wir den »Modul-Steckplatz«, der auch »Erweiterungsanschluß« genannt wird.

Von nahem betrachtet, zeigt der Steckplatz oben und unten je 22 freiliegende Metallanschlüsse (in der Fachsprache »Pins« genannt). Zwischen die Pins kann ein sogenanntes »Modul« gesteckt werden (Bild 17.1).

Bild 17.1: Ein Modul enthält Programme und wird in den Erweiterungs-Port eingesteckt.

Ein Modul ist eine Speichererweiterung. Durch Einstecken eines Moduls in den Erweiterungsanschluß des C 64 werden Informationen (zum Beispiel Programme) »eingebaut«. Wir müssen uns die Sache so vorstellen: Der Speicher des C 64 ist ein Buch mit vielen Seiten. Durch Verwendung von Modulen wird er vergrößert oder verändert (denkt an den Namen »Erweiterungsanschluß«).

Gehen wir die Sache an einem Beispiel durch. Wenn wir auf dem C 64 ein Spiel spielen, haben wir es vorher von einem Datenträger (Diskette oder Kassette) in den Speicher des Computers geladen. Diesen Arbeitsgang können wir uns sparen, wenn das Spiel in Form eines Moduls vorhanden ist. Ein Modul ist ein Stück Computer-Speicher, das wir in die Hand nehmen und nach Belieben ein- und ausbauen können. Unser Spiel wurde vom Hersteller fest in das Modul eingebaut (der Fachmann sagt: »gebrannt«). Spielen wir Computer: »Hey, ich bin eingeschaltet worden. Was soll ich tun? Als erstes muß ich nachschauen, ob in meinem Speicher alles in Ordnung ist. Schnell los. Aha, da ist etwas in den Erweiterungsanschluß gesteckt. Ich muß hingehen und das im eingesteckten Modul befindliche Programm sofort starten.«

Die ganze Sache läuft ruckzuck ab, wir können direkt nach dem Einschalten mit dem Spiel beginnen. Kein stundenlanges Einladen von Programmen mehr. Dieses Modul ist

ein ROM. Ihr erinnert Euch: ROM bedeutet »Nur-Lese-Speicher«. Alle ROMs sind fest im Computer verankert und stehen praktisch ohne Zeitverlust nach dem Einschalten zur Verfügung. Wir müssen das Spiel nicht in den Speicher laden, weil es dort fest verankert ist. Achtung: Eine Sache muß beim Umgang mit Modulen immer beachtet werden. Sie dürfen nur in den ausgeschalteten Computer eingesteckt werden. Wenn Ihr ein Modul in den eingeschalteten Computer steckt, geht er kaputt, also Vorsicht!

Puh, da kommt Verwirrung in meine Speicher-Bits, soviel auf einmal. Ich fasse das neue Wissen kurz zusammen: Module sind »tragbare« Speicher, in die alle Arten von Programmen fest eingebaut werden können (zum Beispiel Spiele oder Dateiverwaltungen). Sie werden in den Erweiterungsanschluß des ausgeschalteten C 64 gesteckt. Ohne Zeitverlust ist das auf dem Modul enthaltene Programm betriebsbereit, da es fest in den Speicher eingebaut ist. Alles klaro? Die erste Etappe der »Tour-de-C 64« haben wir hinter uns.

Ein Stecker, der die Welt bedeutet

Neben dem Modul-Steckplatz befinden sich die Anschlüsse für den Fernseher und der »Audio&Video-Ausgang«, über den unser Computer an eine Stereo-Anlage angeschlossen werden kann. Wenn aus uns später große Computer-Musiker geworden sind, können wir unsere Töne mit diesem Ausgang laut in die Welt hinausposaunen. Die nächsten beiden Ports auf unserer Rundreise dienen der Datasette und der Floppy-Station. Seit dem Drucker-Kapitel verwenden wir den Floppy-Ausgang auch für den Anschluß von Commodore-Druckern.

Wenn wir den C 64 von hinten betrachten, entdecken wir ganz rechts den »User-Port« (»User« heißt auf deutsch »Benutzer«). Das ist ein tolles Ding. Er bietet eine Reihe von Anwendungsmöglichkeiten. Der User-Port unterscheidet sich deutlich vom Modul-Steckplatz. Wir sehen eine »Zunge«, die oben und unten je zwölf Metallkontakte hat (Pins). Acht dieser Metallnippel können nach Bedarf programmiert werden. Okay Freunde, das sagt uns im Moment recht wenig. Was soll das: Pins programmieren?

Der C 64 kann viel mehr als »nur« Spiele spielen oder Adressen verwalten. Pins programmieren bedeutet, wir können über den User-Port Informationen nach »draußen« an andere Geräte geben (Ausgabe), aber auch Informationen erhalten (Eingabe). Bei einem an den C 64 angeschlossenen Gerät wäre eine »Ausgabe« von Informationen zum Beispiel der Befehl: »Gerät einschalten«.

Das Faszinierendste ist folgendes: Über die Programmierung der Pins können bei richtiger Handhabung alle möglichen Dinge »gesteuert« werden. Ein Beispiel: Während andere Eisenbahn-Besitzer mühsam über die Anlage robben, könnt Ihr per Knopfdruck in drei Bahnhöfen das Licht an- oder ausmachen und drei Lokomotiven losfahren lassen. Ihr sitzt einfach am C 64-Kontrollpult und gebt die Befehle über die Tastatur ein.

Sind das nicht bombige Aussichten? Der User-Port hat es faustdick hinter den Ohren. Das probieren wir jetzt aus.

Von Röhren und stürzendem Wasser

Bevor es losgeht, müssen wir einige Dinge über Elektrizität wissen. Warum können wir bei einer Taschenlampe auf Knopfdruck die Birne zum Leuchten bringen?

In der Taschenlampe befindet sich eine Batterie, in der Strom gespeichert ist. Er beginnt zu fließen, wenn wir den Knopf der Taschenlampe drücken und damit die beiden Pole der Batterie verbinden. Zwischen den beiden Polen befindet sich das Birnchen. Diese Tatsache könnt Ihr ganz einfach ausprobieren, indem Ihr Euch eine der großen Batterien besorgt, die man auf den Tisch stellen kann und aus der zwei Drähte ragen. Die beiden Drähte sind die Pole. Wenn Ihr ein Birnchen zwischen die beiden Pole bringt, beginnt es zu leuchten: Der Stromkreis ist geschlossen. Die beiden Pole der Batterie heißen Plus- und Minus-Pol (auf der Seite der Batterie steht, welcher Pol Plus und welcher Minus ist). Durch die Birne fließt ein Strom, der sie zum Leuchten bringt. Stellt Euch den Strom wie einen Fluß aus Wasser vor, der durch die Birne fließt und sie leuchten läßt. Der C 64 wird gleich unsere Batterie sein, denn durch den User-Port besitzt der Computer zwei zugängliche Pole, wie eine Batterie.

Beim Umgang mit den beiden Polen des Computers müssen wir sehr vorsichtig sein. Wenn einfach ein Kabel von einem Pol zum andern gelegt wird, geht der C 64 kaputt. Ein Kurzschluß entsteht. Wenn die beiden Pole der Batterie verbunden werden, ist sie auch in kürzester Zeit im Eimer! Deswegen: Bei allen Arbeiten muß mit größter Sorgfalt zu Werke gegangen werden.

Leider können wir an dieser Stelle nur theoretisch arbeiten, der wirkliche Anschluß eines Birnchens oder zum Beispiel einer Lichtorgel erfordert einige weitergehende elektrotechnische Kenntnisse. Aber keine Sorge, wer nach diesem Kapitel Lust zum Basteln bekommen hat, kann sich ja die nötige Literatur holen und richtig losbasteln.

Zurück zum Thema Strom. Wir müssen uns die Funktion einer Batterie so vorstellen: Wir haben zwei Pole in Form von Röhren. Die beiden Röhren stehen direkt nebeneinander und sind fünf Meter hoch. Die eine Röhre ist bis oben mit Wasser gefüllt, in der anderen Pol-Röhre ist kein Wasser. Jetzt nehmen wir unsere Lampe und bohren in die mit Wasser gefüllte Röhre ein Loch. Das Wasser will durch das entstandene Loch heraus, muß aber durch die Birne fließen. Die Birne leuchtet.

Durch das Leuchten der Birne verliert das Wasser an Kraft. Auf der anderen Seite des Birnchens fließt das Wasser ganz träge in die bisher leere Röhre. Die Batterie ist leer, wenn in der vorher mit Wasser gefüllten Röhre nicht mehr genug Wasser ist. Beide Röhren haben einen etwa gleichhohen Wasserstand.

Wenn wir einen Kurzschluß produzieren, so stürzt das Wasser der vollen Röhre durch ein riesiges Loch mit großer Gewalt in die leere Röhre. Batterie und besonders C 64 überleben diese Aktion nicht.

Es werde Licht!

Wie schon gesagt: Der User-Port bietet die Möglichkeit für Anschluß und Steuerung elektrischer Geräte.

Wir sehen uns Bild 17.2 an: Es zeigt die »Pin-Belegung« des User-Ports. Auf diesem Bild sehen wir den User-Port von hinten. Oben sind die Pins von 1 bis 12 durchnumeriert und unten von A bis N. Von den ganzen Bezeichnungen interessieren uns im Moment nur die Pins A, C, D, E, F, H, J, K und L. Erinnert Ihr Euch an die beiden Pole der Batterie? Wir werden Pin A zu unserem Minus-Pol und Pin C zum Plus-Pol machen. Die Sache ist ganz einfach. Pin A ist vom Computer als Minus-Pol festgelegt. Unter der Spalte »Belegung« steht für A »GND«. Das ist die englische Abkürzung für »Ground« (auf deutsch »Boden, Erdung«) und steht für »Minus-Pol«. Pin C machen wir mit zwei kleinen Befehlen zum Plus-Pol.

1 2 3 4 5 6 7 8 9 10 11 12

A B C D E F H J K L M N

PIN	BELEGUNG	PIN	BELEGUNG
1	GND	A	GND
2	+5V, max.100 mA	B	FLAG2
3	$\overline{\text{RESET}}$	C	PB0
4	CNT1	D	PB1
5	SP 1	E	PB2
6	CNT2	F	PB3
7	SP 2	H	PB4
8	$\overline{\text{PC2}}$	J	PB5
9	SER. ATN IN	K	PB6
10	9V AC, max. 100 mA	L	PB7
11	9V AC, max. 100 mA	M	PA2
12	GND	N	GND

Bild 17.2: Pin-Belegung des User-Ports.

Kommen wir zum Programmieren der einzelnen Pins. Wir haben im Kapitel über Speicher und binäre Zahlen gelernt: Ein Byte besteht aus acht Bit, die die Potenzen der Zahl 2 enthalten.

Die Zimmerbelegung eines Bytes wird mit den Zahlen null und eins dargestellt. Die Eins steht für: Bit gesetzt, die Null für: Bit nicht gesetzt. Wir können über ein bestimmtes Byte die acht Pins C bis L nach Belieben programmieren. Eine Eins bedeutet: Pin auf Ausgang gesetzt, eine Null: Pin auf Eingang gesetzt. Die benötigte Speicherstelle ist das Datenrichtungsregister-Byte mit der Nummer 56579.

Jedes Bit in dieser Speicherstelle steht für einen Pin (von C bis L) des User-Ports. Halt, halt, halt. Nicht so schnell. Meine Ohren haben sich automatisch zugeklappt, als plötzlich die ganzen Informationen da oben auf mich einprasselten. Die Pins wirbeln in meinem Schädel herum und stoßen sich mit Byte 56579. Nun mal langsam.

Bitchen für Bitchen Qualität!

Der User-Port hat 24 Pins. Von diesen 24 Pins werden acht (Pin C bis L) durch die Speicherstelle 56579 dargestellt. Dieses Byte wird »Datenrichtungsregister« genannt, weil in ihm festgelegt wird, welcher Pin Daten aussendet (auf eins gesetzt) und welcher Daten empfängt (auf null gesetzt). Es legt fest, in welche Richtung die Daten fließen: rein oder raus.

Überprüfen wir unsere Kenntnisse an einem Beispiel. Wir schalten den C 64 an und tippen ein:

```
PRINT PEEK (56579)
```

Es erscheint eine Null. Das bedeutet: Nach dem Einschalten steht im Datenrichtungsregister die binäre Zahl 00000000. Jeder Pin von C bis L ist auf »Eingang« gesetzt. Pin C entspricht Bit 0 (=2^0, das ist die Null ganz rechts) und Pin L steht für Bit 7 (=2^7, die Null ganz links).

Bild 17.3 zeigt die Pin-Belegung nach EinPOKEn der Zahl 7 (binär 00000111): Die Pins C bis E sind auf Ausgang gesetzt, der Rest auf Eingang.

```
POKE 56579,7
```

erledigt diese Arbeit für uns.

auf Eingang

L K J H F E D C

auf Ausgang

Speicherstelle 56579							
0	0	0	0	0	1	1	1
dezimal: 7							
POKE 56579, 7							

Bild 17.3: Die Darstellung der Pin-Belegung des Datenrichtungsregisters nach EinPOKEn von 7.

Ein auf »Eingang« gesetzter Pin empfängt Informationen von außen, während nur die auf »Ausgang« gesetzten Pins die Voraussetzung für den Befehl zum Anschalten eines Birnchens sind.

Die Pins C, D und E sind durch Setzen der ersten drei Bits des Datenrichtungsregisters auf Ausgang geschaltet.

Fassen wir an dieser Stelle kurz zusammen. Wir sind Besitzer einer Modelleisenbahn. Aus diesem Grund möchten wir die Grundlagen kennenlernen, mit denen der C 64 in einem bestimmten Bahnhof oder unabhängig davon auf einem Bahnsteig das Licht anmachen kann. Zu diesem Zweck brauchen wir wie bei einer Batterie einen Plus- und einen Minus-Pol. Der Minus-Pol ist uns bereits gegeben: Pin A des User-Ports. Wo aber sind die Plus-Pole? Sie bauen wir uns mit Hilfe des Datenrichtungsregisters 56579. Es kontrolliert mit seinen acht Bit die acht Pin C bis L. Eben haben wir die Pins C, D und E auf Ausgang gesetzt. Nehmen wir an, daß Pin C für die Beleuchtung unseres Bahnhofs zuständig ist und Pin D für die des Bahnsteigs. Bisher haben wir sozusagen die Licht-Leitungen gelegt. Es fehlt uns ein Schalter, der das Licht an- oder ausmacht. Das ist genauso wie bei der Taschenlampe. Sobald sich eine Batterie in der Taschenlampe befindet, kann sie eingesetzt werden. Licht entsteht aber erst, wenn wir den Schalter der Taschenlampe auf »ein« stellen. Klar?

Licht an und aus

Der Ein- und Ausschalter ist die Speicherstelle 56577. Dieses Byte wird auch »Datenregister« genannt. Es kann Pins, die im Datenrichtungsregister auf Ausgang geschaltet worden sind, ein- oder ausschalten.

Das bedeutet, mit ihm können wir das Licht im Bahnhof einschalten. Verschaffen wir uns einen kleinen Überblick.

```
PIN           L K J H F E D C
binäre Zahl   0 0 0 0 0 1 1 1(=7 dezimal)
(Byte 56579)
```

Pin C, D und E sind durch das Datenrichtungsregister auf Ausgang gestellt. An Pin C hängt die Beleuchtung des Bahnhofs, an D die des Bahnsteigs.

Die Bitbelegung des Datenregisters 56577 sieht im Moment noch so aus:

```
binäre Zahl   0 0 0 0 0 0 0 0(=0 dezimal)
(Byte 56577)
```

Nichts passiert, weil kein einziger Pin eingeschaltet ist. Unser Ziel ist es, im Bahnhof an Pin C Licht anzumachen. Zu diesem Zweck müssen wir das erste Bit der Speicherstelle 56577 setzen.

Binär stellt sich das folgendermaßen dar:

```
PIN           0 0 0 0 0 0 0 1(=1 dezimal)
(Byte 56577)
```

Der Befehl zum Einschalten von Pin C wäre:

```
POKE 56577,1
```

Im Bahnhof geht das Licht an. Sehen wir uns Bild 17.4 an. Es zeigt unsere bisherigen Aktivitäten auf einen Blick.

Sprite-Nummern zugehörige Werte

128 64 32 16 8 4 2 1

7 6 5 4 3 2 1 0

0	0	0	0	0	1	0	0

Eine 1 für den gewünschten Sprite

Bild 17.4: Eine Übersicht der verschiedenen Eingaben.

Ganz oben sind die Pins aufgelistet, darunter die beiden Register 56579 und 56577. Beide Register stellen mit ihren Bits die einzelnen Pins dar.

Zuerst haben wir im Datenrichtungsregister mit POKE 56579,7 die ersten drei Pins auf Ausgang geschaltet. Pin C steuert das Licht des Bahnhofs, D das des Bahnsteigs. Als nächstes haben wir uns das Datenregister 56577 vorgenommen. Um Pin C anzuschalten (und damit das Licht im Bahnhof), müssen wir in das Datenregister den Wert der binären Zahl 00000001(=1 dezimal) eingeben. Das haben wir durch POKE 56577,1 getan: Im Bahnhof geht das Licht an. Das ist eine tolle Sache was? Ihr sitzt einfach am Computer, gebt einen bestimmten Befehl ein, und weit entfernt von Euch geht geisterhaft das Licht an.

Heller Bahnsteig

Auf genauso geisterhafte Weise geht das Licht wieder aus. EinPOKEn von

```
POKE 56577,0
```

setzt das erste Bit wieder zurück. Damit nicht genug, wir wollen jetzt die Beleuchtung auf dem Bahnsteig einschalten. Zu diesem Zweck müssen wir das zweite Bit im Datenregister einschalten. Die binäre Zahl hierfür ist

```
binäre Zahl  0 0 0 0 0 0 1 0(=dezimal 2)
(Byte 56577)
```

Die Eingabe von

```
POKE 56577,2
```

schaltet Pin D auf »Ein« und damit das Licht auf dem Bahnsteig an. Das Datenregister-Byte ist also der elektronische An- und Ausschalter der in der Umwelt verstreuten Befehlsempfänger des C 64.

Mannomann, da kommen mir jede Menge Ideen, was ich mit dem Computer anstellen kann. Stellt Euch eine Party mit einer C 64-gesteuerten Lichtorgel vor!

Zuckersüße Programmier-Erfolge

Ist das nicht fantastisch? Der C 64 bietet eine Vielzahl von interessanten Anwendungsmöglichkeiten. Wir haben eben in die Grundlagen eines bisher völlig unbekannten Gebietes hineingeschaut: Messen – Steuern – Regeln. Dieser kleine Ausflug führt leicht in sehr wissenschaftliche Gebiete. Wie schon erwähnt, kann zum Beispiel der C 64 sinnvoll für den Umweltschutz eingesetzt werden und radioaktive Strahlung messen. Er hilft in allen möglichen Gebieten der Technik: zum Beispiel als Steuerungscomputer von Fräsmaschinen oder Werkbänken.

Manch ein Bastel-Freak hat seinen C 64 in ein mit Knöpfen, Schaltern, Kabeln und Lämpchen überzogenes Elektronik-Wunder verwandelt. Wer gerne bastelt und Interesse für die Elektronik hat, ist hier an der richtigen Adresse. Ich muß gestehen, mich juckt es in den Fingern, wenn ich diese tollen und faszinierenden Geräte sehe.

Große Entdeckungen

Ganz am Anfang haben wir uns durch Basic gewühlt, dann kam der Speicher mit seinen für bestimmte Informationen reservierten Registeradressen (wir haben eben weitere reservierte Adressen kennengelernt, zum Beispiel das Datenrichtungsregister 56579!). Jetzt steht unser ewig wissensdurstiger Fuß bereits weit in der neuen Welt von Messen, Steuern und Regeln.

Erinnert Ihr Euch an unsere Anfänge, wie schwierig der Anschluß des Fernsehers und die ersten Computer-Schritte waren? Mittlerweile beherrschen wir den Kasten hervorragend und können ihn für tolle Sachen verwenden! Alles geht leicht von der Hand, wenn man zäh und spaßig ist. Das sind wir, und nicht zu knapp.

Das haben wir gelernt

User-Port: Eine wichtige Anschluß- und Schnittstelle des C 64. Durch ihn kann der C 64 Daten empfangen, aber auch an andere Geräte schicken. Mit zwei POKEs können wir den User-Port programmieren. Mit einem definieren wir, welche Pins Ausgänge und welche Eingänge sind. Mit dem zweiten POKE bestimmen wir, an welchen Ausgängen tatsächlich Spannung liegt.

Pins: Die Kontakte oder Metallzungen des User-Ports.

Datenrichtungsregister: Bestimmt in welche Richtung Daten ausgetauscht werden, also vom C 64 nach außen, oder von außen nach »innen«.

Datenregister: Setzt Ausgabeleitungen. Durch Veränderung der Werte in diesem Register bestimmen wir, an welchen Ausgängen des User-Ports sich Strom befindet.

Kapitel 18

Raumschiffe, grüne Männchen und vor allem zwei Auto-Sprites

Schwarz und matt liegt das Auge des Computers da, nichts auf dem Bildschirm zeigt Leben. Aus dem Nichts taucht eine Hand auf und erweckt mit zwei kleinen Bewegungen Bits und Bytes zum Leben. Eine Wüste entsteht: Die gefürchtete, staubtrockene »Sierra Commodore«. In dieser Wüste ist schon manch ein Programmierer steckengeblieben. Hoch oben am Horizont leuchtet das allwissende

```
*** COMMODORE 64 BASIC V2 ***
```

und sein Helfershelfer, der blinkende Cursor. Plötzlich wieder die Hand, einige kleine Programmierbewegungen verändern alles. Der Horizont macht blauem Flimmern Platz, eine Straße führt durch die Wüste. Am Straßenrand steht ein Haus, aus rohen Balken gezimmert. Links am Blickfeldrand erscheint ein weißes Cabriolet, von rechts bahnt sich ein riesiger schwarzer Lastwagen seinen staubigen Weg...

Hey, Leute, Ihr habt richtig verstanden. Die »Sierra Commodore« ist der Bildschirm und die programmierende Hand unsere eigene. Die zwei Autos und das Haus stammen von uns. Ein neues Programm machts möglich: Die »Sprite-Programmierung«.

Sprite ungleich Sprite

Klar, das Wort »Sprite« hat nichts mit Limonade zu tun. Es ist der Oberbegriff für alles, was auf dem Bildschirm kreucht und fleucht, fliegt und flattert. Die kleine Giana aus »The Great Giana Sisters« ist ein Sprite, genauso wie die Diamanten und hüpfenden Ungetiere, gegen die sie kämpft. Sprites sind demnach von uns definierte Figuren beziehungsweise Zeichnungen, die wir über den Bildschirm bewegen können, ohne den Hintergrund zu zerstören. Grafiken, die mit dem PRINT-Befehl erzeugt werden, zerstören den Hintergrund, da sie den Bildschirminhalt einfach überschreiben. Erinnert Ihr Euch an das Programm, in dem wir einen Ball über den Bildschirm bewegt haben? Dort mußten wir ständig Bildschirmpositionen überschreiben, um eine Bewegung zu simulieren. Sprites hingegen legen sich quasi über den Bildschirm, aber zerstören nicht

den Hintergrund, da nichts überschrieben werden muß. Hinter den Sprites steckt eine bestimmte Programmierung. Diese werden wir uns gleich näher anschauen und sie an einigen Beispielen ausprobieren. Hinein in das Vergnügen.

Zuerst wird gemalt. Wir können unserem Sprite jede beliebige Form geben, soll es ein kleines Männchen, ein Raumschiff, eine Blume oder ein Auto sein? Wir wählen ein Auto. An diesem gehen wir den Werdegang eines Sprites durch.

Das Grundprinzip ist folgendes: Die Gestalt der Figur wird in Zahlen übertragen, die im Speicher des Computers abgelegt werden. Sie sind der Grundstock, auf dem wir aufbauen.

Für die Darstellung des Sprites auf dem Bildschirm benötigt der C 64 zusätzliche Informationen. Zum Beispiel:

1. Wo befinden sich die Zahlen, die das Aussehen festlegen (Speicherstellen)?
2. Welche Farbe hat es (16 verschiedene Möglichkeiten!)?
3. Ist es groß oder klein?
4. An welcher Stelle auf dem Bildschirm soll es erscheinen?

Für jede Information gibt es eine bestimmte Speicherstelle. In einer befindet sich die Farbe des Sprites, in einer anderen die Bildschirmposition (links oben in der Ecke, unten in der Mitte oder irgendwo dazwischen).

Freunde, von der Konstruktion des Autos bis zum entgegenkommenden schwarzen Lastwagen machen wir alles selber. Wir wandeln unser Auto in Zahlen um. Alles, was wir auf dem Bildschirm sehen, besteht aus kleinen, aneinandergereihten Punkten. Ein »A« zum Beispiel ist ein leuchtender Haufen von Punkten, der die Form »A« hat.

Wenn Ihr ganz nah an den Bildschirm herangeht, könnt Ihr sie unterscheiden. Genau das gleiche Punkt-Prinzip steckt hinter der Konstruktion von Sprites. Sie sind nichts anderes als ein Haufen Leuchtpunkte, dessen Form wir bestimmen. Wir können sagen: »Hey C 64, diese Punkte sollen leuchten und diese nicht«. Ein Schema nimmt uns die meiste Arbeit ab. Es besteht aus 24 Spalten und 21 Reihen.

Die dadurch entstehenden Punkte können wir einzeln an- und ausmachen. Bild 18.1 zeigt unser erstes Beispiel, das Cabriolet.

Spalten-nummer	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	Zahlencodes		
Werte	128	64	32	16	8	4	2	1	128	64	32	16	8	4	2	1	128	64	32	16	8	4	2	1	1	2	3
Zeile 1																									0	0	0
Zeile 2																									0	0	0
Zeile 3																									0	0	0
Zeile 4																									0	0	0
Zeile 5																									0	0	0
Zeile 6																									0	0	0
Zeile 7																									0	0	0
Zeile 8																									0	0	0
Zeile 9																									0	0	0
Zeile 10																									0	0	0
Zeile 11																									0	0	0
Zeile 12																									0	3	0
Zeile 13																									0	0	128
Zeile 14																									0	0	64
Zeile 15																									127	255	254
Zeile 16																									127	255	252
Zeile 17																									120	127	14
Zeile 18																									123	127	110
Zeile 19																									3	0	96
Zeile 20																									0	0	0
Zeile 21																									0	0	0

Bild 18.1: Das Konstruktionsblatt für Sprites: Die ausgefüllten Kästchen werden in Zahlen umgerechnet und in Bytes eingesetzt.

Schwarzes Zahlenauto

Für den Anfang malen wir das Konstruktionsschema auf ein großes, kariertes Blatt Ringbuchpapier. 24 senkrechte »Spalten« und 21 waagrechte »Reihen«. Unser Sprite setzt sich aus 24x21 = 504 einzeln steuerbaren Punkten zusammen. In Bild 18.1 habe ich alle die Punkte schwarz ausgemalt, die das Cabriolet ergeben. Wir müssen aufpassen: Jeder schwarze Punkt wird auf dem Bildschirm dargestellt. Halbe Felder gibt es nicht, entweder sie sind schwarz oder leer. So, Leute, jetzt kommt ein Trick. Die schwarzen Felder in Bild 18.1 ergeben die neben dem Schema angegebenen Zahlen.

Halt, halt, jetzt mal langsam. Sehen wir uns die Zahlen oberhalb des Schemas in Bild 18.1 etwas genauer an. Es sind drei Reihen, die von 128 bis 1 hinuntergehen. Die Zahlen sind die bei der Bit-Programmierung kennengelernten Potenzen von 2. Rechts von den Zahlen gibt es drei Spalten mit den Überschriften »1.Byte, 2.Byte, 3.Byte«. Da dämmert es. Ein Byte besteht aus acht Bit, die die Potenzen der Zahl Zwei darstellen. Ein Byte enthält eine Zahl zwischen 0 und 255, das ist uns bekannt. Wenn wir auf dem Zettel ein Kästchen schwarz malen, »setzen« wir das entsprechende Bit (die Zahl, die über der Spalte steht). Das Setzen des Bits »schaltet« den entsprechenden Abschnitt des Sprites »an«. Am Ende zählen wir die Werte der einzelnen Bits zusammen und schreiben das Ergebnis in die richtige Spalte auf der rechten Seite des Schemas. Jede waagrechte Zeile besteht aus drei Byte (die senkrechten dunklen Linien geben die Grenzen zwischen den Byte an). Das Ergebnis des ersten Bytes (die erste Reihe von 128 bis 1) tragen wir in die Spalte mit der Überschrift »1.Byte«, »2.Byte« nimmt die Summe der zweiten Zahlenreihe auf und »3.Byte« die der acht Zahlen ganz rechts.

Links oben gibt es in der ersten Bit-Reihe von 128 bis 1 kein einziges schwarzes Kästchen. Wir brauchen nicht zu rechnen, in jedem Bit steht Null. Das Ergebnis schreiben wir in die erste Position der Spalte »1.Byte«: 0. Die erste Position von »2.Byte« bekommt ebenfalls eine Null, da der mittlere Abschnitt der obersten Kästchenreihe auch kein ausgefülltes Kästchen enthält. Nach diesem Schema gehen wir die gesamte Zeichnung durch. Anfangs tragen wir überall Nullen ein, bis wir in der mittleren Byte-Spalte auf das Dach des Cabriolets treffen. Hier müssen wir eine 3 unter »2.Byte« eintragen, weil die Kästchen (der Bits) für Zwei und Eins ausgefüllt sind. Wenn wir das Reihe für Reihe durchgehen, erhalten wir die in Bild 18.1 angegebenen Zahlen. Unser Auto ist in insgesamt 63 Zahlen zerlegt. Jetzt werden diese Zahlen in den Speicher des C 64 gepackt.

Byte für Byte Qualität

Erinnert Ihr Euch an den Bildschirmspeicher? Der Bildschirmspeicher ist ein genau abgegrenzter Bereich, in dem die dargestellten Zeichen abgelegt sind. Das Speichern der 63 Zahlen eines Sprites verläuft nach dem gleichen Prinzip. Der Sprite-Speicher beginnt bei Byte Nummer 704. Für uns bedeutet das: Wir legen die 63 Zahlen des Cabriolets in die Speichernummern 704 bis 766 ab. Die benötigte Horde Möbelpacker, die das Zahlenauto in die richtigen Speicherstellen räumt, bietet der POKE-Befehl. Legen wir los:

```
POKE 704,0
POKE 705,0
POKE 706,0
```

»Gähn«, wenn wir bis zum bitteren Ende so weitermachen, haben wir einen Bart, der sich zweimal um den Computer wickeln läßt. Es muß einen schnelleren und eleganteren

Weg geben: Eine FOR-NEXT-Schleife mit dem READ-DATA-Befehl ist die Lösung. Die Zahlen werden in DATA-Zeilen gepackt und der C 64 liest jede Zahl in seinen Speicher ein. Bevor wir richtig herumPOKEn, strengen wir unser Gehirn ein wenig an. Nur der Dumme macht sich zuviel Arbeit. Wir müssen die Nullen nicht alle eingeben. In den unbenutzten Sprite-Speicherstellen ist von vorneherein eine Null gespeichert. Das überprüfen wir mit

```
PRINT PEEK (730)
```

Es erscheint eine Null. Für uns bedeutet das: Die Nullen am Anfang unserer Sprite-Zahlen können weggelassen werden. Jetzt müssen wir ein wenig aufpassen, damit der Sprite-Speicher nicht durcheinanderkommt. Die ersten 30 Nullen können wir weglassen. Das bedeutet: Das EinPOKEn der ersten 30 Werte entfällt, die letzten sechs Nullen (die Zeilen unter den Rädern) sind auch unnötig. Damit geht der von uns benötigte Sprite-Speicher von 734 bis 760.

Fassen wir kurz zusammen. Ein Sprite wird in einem Raster in Zahlen verwandelt, indem einzelne Bits an- oder ausgeschaltet werden. Die Ergebnisse (63 Zahlen) werden im Sprite-Speicher ab Byte Nummer 704 eingetragen. Nullen müssen nicht eingegeben werden, da leere Bytes eine Null beinhalten. Aus diesem Grunde schränken wir den einzuPOKEnden Bereich ein: Die unwichtigen Bytes werden übergangen. In der Tabelle zählen wir bis kurz vor dem Dach unseres Autos:

```
703  704  705
706  707  708
709  ...  733
734  735  736
737  ...  760
761  762  763
764  765  766
```

Bei Byte 734 beginnen wir mit dem EinPOKEn und hören bei 760 auf. Alles klaro? Dann zurück zur FOR-NEXT-Schleife!

Ein Auto wird gePOKEt

Wir benötigen eine Befehlsfolge, die die 27 übrigbleibenden Auto-Werte in die Speicher-Bytes 734 bis 760 einPOKEt. Was haltet Ihr von folgender Idee:

```
280 FOR N=0 TO 26:READ Q
    :POKE 734+N,Q:NEXT
370 DATA 0,0,0,0,3,0,0,0,128,0,0,64
380 DATA 127,255,254,127,255,252,
```

```
    120,127,14,123,127,110
390 DATA 3,0,96
```

Laßt Euch von den hohen Zahlen der Befehlszeilen nicht verwirren, die gehören schon zu unserem Lastwagen-Cabriolet-Programm.

Zeile 280 gibt dem C 64 mehrere Befehle: Zähle für die Variable N von Null bis 26, beginne mit N=0. Lies aus den DATA-Zeilen den ersten Wert und lege ihn unter der Bezeichnung Q in deinen Speicher. POKE 734+0,0 (die erste Null ist der momentane Wert für N, die zweite der erste ausgelesene Wert von Q). Zähle dann das nächste N (diesmal N=1), lies den nächsten Q-Wert aus DATA und POKE 704+1,0. Zähle N=2 und so weiter. Der C 64 liest die 27 Werte aus den DATA-Zeilen und POKEt sie in den Speicher. Zur besseren Übersicht packe ich in jede DATA-Zeile höchstens 12 Werte.

Leute, unser Auto befindet sich im Speicher, zu sehen ist aber noch nichts. Bevor wir es auf dem Bildschirm sehen können, müssen wir einige Dinge bedenken. Der C 64 kann bis zu acht Sprites auf einmal produzieren. In unserem Video-Spiel »Giana« tauchen neben Giana Tiere und Diamanten auf. Wo befindet sich der Speicher für diese neuen Sprites? Er liegt hinter dem uns bekannten Sprite-Speicher. Jedes Sprite benötigt 63 Byte als Speicherplatz. Tabelle 18.1 zeigt einige Adressen der einzelnen Speicher.

Speicherbereich	Codezahl
704-766	11
832-894	13
896-958	14
960-1022	15

Tabelle 18.1: Speicherbereiche zum Ablegen von Sprite-Daten.

Jeder Abschnitt des Sprite-Speichers hat eine eigene Nummer. Sie teilt dem Computer den Bereich mit, in dem das Sprite plaziert werden soll. Stellt Euch vor, Ihr seid ein Computer. Plötzlich kommt einer an, schreibt Zahlen in Euren Speicher und befiehlt: »Lies die Zahlen und zeichne ein Sprite auf den Bildschirm!«, sagt Euch aber nicht, wo die Zahlen zu finden sind. Zu diesem Zweck gibt es verschiedene Code-Zahlen, die dem Computer den Weg weisen. Tabelle 18.1 zeigt ein paar Beispiele.

Wir haben die Zahlen in den Speicher mit der Code-Zahl 11 eingetragen. Diese Zahl wird in Byte 2040 abgelegt.

```
240 POKE 2040,11
```

Das Sprite befindet sich im Speicher und der C 64 weiß durch Zeile 240, wo er das Sprite finden kann. Das Cabriolet-Sprite erhält die Nummer 0. Wir können die Code-Zahl für den Speicherbereich in eins der Bytes 2040 bis 2047 legen (Bild 18.2).

2040	41	42	43	44	45	46	2047
↑	↑	↑	↑	↑	↑	↑	↑
Sprite 0	1	2	3	4	5	6	7

Bild 18.2: Die Sprites und ihre Speicherstellen.

Wenn wir

```
POKE 2044,11
```

eingeben, so geben wir dem Sprite die Nummer 4, 2045 wäre Sprite Nummer 5. Was das bedeutet, probieren wir später aus. Wir müssen uns im Moment merken:

Der C 64 kann acht Sprites gleichzeitig auf seinem Bildschirm darstellen. Für diese acht Sprite gibt es verschiedene Speicherbereiche mit je 63 Speicherstellen (jedes Sprite wird in 63 Zahlen zerlegt). Die Speicherbereiche haben jeder eine eigene Code-Zahl, mit der der C 64 den richtigen Abschnitt anfährt. Die Code-Zahl (in unserem Beispiel »11«) wird in eins der Bytes von 2040 bis 2047 eingePOKEt. Damit legen wir zwei Dinge fest: Erstens kennt der Computer die Adresse des Sprites und zweitens legen wir mit der Wahl des Bytes (2040 enthält die Code-Zahl für Sprite 0, 2047 die für Sprite 8) die Nummer des Sprites fest. Wir haben uns für Sprite Nummer 0 entschieden, da

```
POKE 2040,11
```

eingegeben wurde. So, Freunde, jetzt geht es rund. In einer Blitzaktion rufen wir das Cabrio-Sprite auf den Bildschirm. Gebt ein:

```
250 POKE 53269,1
310 POKE 53248,50:POKE 53249,100
```

Wir starten das Programm mit

```
RUN
```

Auf dem Bildschirm erscheint ein wunderschönes schneeweißes Cabriolet. Ein siebenzeiliges Programm ruft ein fesches Auto auf den Bildschirm.

```
240 POKE 2040,11
250 POKE 53269,1
280 FOR N=0 TO 26:READ Q
    :POKE 734+N,Q:NEXT
310 POKE 53248,50:POKE 53249,100
370 DATA 0,0,0,0,3,0,0,0,128,0,0,64
```

```
380 DATA 127,255,254,127,255,252,
    120,127,14,123,127,110
390 DATA 3,0,96
```

Die neuen Zeilen sind 250 und 310. In Zeile 250 geben wir die Order: Schalte Sprite 0 an. Die Sache funktioniert wie ein Lichtschalter, 1 heißt an, 0 aus. Wenn Euch beim Arbeiten mit dem Programm das Sprite auf dem Bildschirm stört, schaltet es mit POKE 53269,0 aus. Zeile 310 gibt die Position des Sprites auf dem Bildschirm an. Wo soll es erscheinen? Das Byte 53248 nimmt den Wert für die X-Achse auf (waagrechte Position, links oder rechts?), Nummer 53249 bestimmt die Y-Achse (senkrechte Position, oben oder unten?). Hier ein kleiner Vorgriff: Tabelle 18.2 listet die verschiedenen Bytes der Reihe nach auf.

Beide Werte laden zum Ausprobieren ein. Gebt zum Beispiel statt 50 in Zeile 310 eine 10 ein, das Auto ist nur noch zur Hälfte zu sehen, bei Null ist es ganz hinter dem Bildschirmrand verschwunden. Wir sind da einer wichtigen Sache auf der Spur. Sprites können aus dem sichtbaren Bereich des Bildschirms verschwinden und wieder auftauchen. Eine tolle Vorstellung.

Wachs in den Händen

Leute, mit dem fertigen Sprite im Speicher haben wir jede Menge Veränderungsmöglichkeiten. Das kleine Programm lädt zum Spielen und Erweitern ein. Das System ist immer das gleiche. Jede Veränderung des Sprites wird in einem reservierten Speicher niedergeschrieben. Es gibt zum Beispiel ein reserviertes Byte, die Nummer 53287, für die Farbe des Sprites an. Baut diese Zeile in das Programm ein.

```
260 POKE 53287,1
```

Anstelle der Eins können die Zahlen 0–15 eingesetzt werden. Das Sprite bekommt auf Knopfdruck eine neue Farbe. Das Auto kann in X- oder Y-Richtung vergrößert werden. X-Richtung bedeutet: Das Auto wird länger; Y-Richtung bedeutet: Der Wagen wird höher. Wenn beide Vergrößerungen »eingeschaltet« werden, erscheint ein echtes Riesen-Cabrio. So wird es gemacht: Für die X-Achse ist Speicherstelle 53277 zuständig. Wenn hier statt der Null eine Eins steht, streckt sich das Auto. Die Y-Achse übernimmt Byte 53271. Enthält dieses Byte eine Eins, wächst das Cabrio »gen Himmel«. Fügen wir die beiden Zeilen in das Programm ein. Es sieht jetzt so aus:

```
240 POKE 2040,11:REM SPRITE 0
                 MIT CODE-ZAHL 11
250 POKE 53269,1:REM SPRITE 0 AN
260 POKE 53287,0:REM FARBE SPRITE 0
280 FOR N=0 TO 26:READ Q
    :POKE 734+N,Q:NEXT
```

```
310 POKE 53248,50:POKE 53249,100
    :REM KOORDINATEN SPRITE 0 (X,Y)
330 POKE 53277,0:REM X-ACHSE GROESSER
340 POKE 53271,0:REM Y-ACHSE GROESSER
370 DATA 0,0,0,0,3,0,0,0,128,0,0,64
380 DATA 127,255,254,127,255,252,
    120,127,14,123,127,110
390 DATA 3,0,96
```

Ein riesiges Feld von Trainingsmöglichkeiten. Je öfter wir die Werte verändern, desto besser haben wir das Programm im Griff. Jetzt stellen wir uns eine neue Aufgabe: Das Cabriolet soll über den Bildschirm fahren.

Sprite-Rallye

In Zeile 310 liegt der Schlüssel zum Erfolg. Das Auto soll waagrecht über den Bildschirm fahren, der Wert für die X-Achse muß verändert werden. Zu diesem Zweck führen wir eine neue Variable ein: X (anstelle der 50 in Befehl 310). Zusätzlich fügen wir zwei Zeilen ein, die für X immer wieder neue Werte einsetzen.

```
300 FOR X=0 TO 255
310 POKE 53248,X:POKE 53249,100
    :REM KOORDINATEN SPRITE 0
350 NEXT X
```

Durch die Zeilen 300 und 350 zählt der C 64 für X von Null bis 255. Die wechselnden Werte setzt er für X in Befehl 310 ein und verändert die Position des Autos auf der X-Achse. Starten wir die erweiterte Programm-Version mit RUN: Langsam und gleichmäßig düst das Auto von links über den Bildschirm. So einfach ist das. Auf unsere Erfolge können wir stolz sein! Mein über den Bildschirm fahrendes Auto gefällt mir sehr, ist aber noch ausbaufähig. Ich nehme eine neue Zeile in mein Programm auf. Das Cabriolet soll immer wieder von links losfahren, also eine Endlos-Schleife.

```
360 GOTO 300
```

Nach einer Weile habe ich mich sattgesehen. Eine neue Aufgabe wartet. Ein zweites Sprite muß her. Ich will meine neuen Kenntnisse auf eine harte Probe stellen und einen großen Lastwagen programmieren, der dem Cabrio entgegenkommt. Erst dann zeigt sich, ob ich alles richtig verstanden habe. Zunächst nehme ich mir wieder ein Konstruktions-Blatt und zeichne den gewünschten Lastwagen. Er hat einen Kran und transportiert eine schwere Last (Bild 18.3). Rechts trage ich die Werte in die Byte-Spalte ein.

Spaltennummer: 1 2 | 3 4 | 5 6 | 7 8 | 9 10 | 11 12 | 13 14 | 15 16 | 17 18 | 19 20 | 21 22 | 23 24

Werte: 128 64 | 32 16 | 8 4 | 2 1 | 128 64 | 32 16 | 8 4 | 2 1 | 128 64 | 32 16 | 8 4 | 2 1

	Zahlencodes 1	2	3
Zeile 1	0	0	0
Zeile 2	0	0	0
Zeile 3	0	0	0
Zeile 4	0	0	0
Zeile 5	0	0	0
Zeile 6	0	0	0
Zeile 7	0	0	0
Zeile 8	0	0	56
Zeile 9	0	0	68
Zeile 10	7	224	130
Zeile 11	4	33	128
Zeile 12	4	33	0
Zeile 13	63	225	0
Zeile 14	63	225	28
Zeile 15	127	255	28
Zeile 16	127	255	254
Zeile 17	97	254	2
Zeile 18	109	254	218
Zeile 19	12	0	218
Zeile 20			
Zeile 21			

Bild 18.3: Das Konstruktionsblatt des Lastwagens.

Die neuen Sprite-Werte lege ich im Speicher mit der Code-Zahl 13 (Tabelle 18.1) ab. Er geht von 832 bis 894. Für die Daten des Lasters brauchen wir nur die Bytes von 853 bis 888. Hier tauchen einige Neuheiten auf.

Ich schaue mir noch einmal Bild 18.2 an. Welche Nummer soll mein Sprite haben? Die Null ist belegt, ich entscheide mich für die Nummer eins mit Speicherstelle 2041. In meinem Programm spielen die Sprites mit den Nummern null und eins eine Rolle. Nach altbekanntem Muster ergänze ich Zeile 240 mit den neuen Informationen.

```
240 POKE 2040,11:POKE 2041,13
    :REM SPRITE 0 UND 1
```

Der C 64 kennt sowohl die Speicherstelle als auch die Bezeichnung der im Spiel befindlichen Sprites. Er bringt sie nie durcheinander.

Ein weiteres Problem wartet auf seine Lösung: Ich muß die READ-DATA-Befehle sinnvoll in das Programm einfügen. Zeile 280 enthält den POKE-Befehl für das Cabrio,

in die nächste Zeile (290) baue ich das Gerüst für den Lastwagen ein. Das Prinzip ist das gleiche wie beim Cabriolet, nur die Werte sind etwas verschieden.

```
290 FOR T=0 TO 35:READ P
    :POKE 853+T,P:NEXT
```

Die DATA-Informationen packe ich hinter die DATA-Werte des Autos.

```
400 DATA 0,0,56,0,0,68,7,
    224,130,4,33,128
410 DATA 4,33,0,63,225,0,63,
    225,28,127,225,28
420 DATA 127,255,254,97,
    254,2,109,254,218,12,0,218
```

Der C 64 liest nie die falschen Werte. Er beginnt in Zeile 280 und liest die 26 Werte für das Cabriolet, dann springt er in Zeile 290 und holt sich die Werte des Lastwagens, kein Problem für ihn.

Der Brummi fährt los

Die dicke Kiste muß die gleichen Arbeitsgänge durchlaufen wie vorhin unser kleiner Flitzer. Der Anschaltknopf für alle Sprites liegt in der Speicherstelle 53269. Für das Anschalten des Lastwagens müssen wir uns die Vorgänge in diesem Byte näher ansehen. Wir wissen, jedes Byte enthält acht Bit. Gleichzeitig stehen uns bis zu acht Sprites zur Verfügung, die einzeln an- und ausgeschaltet werden können. Die Sprites werden durch Setzen der einzelnen Bits in Speicherstelle 53269 an- und ausgeschaltet. Das Verfahren zeigt uns Bild 18.4.

Speicherstelle 53269

128	64	32	16	8	4	2	1
0	0	0	0	0	0	1	1

Bild 18.4: Die acht Bit sind die Anschaltknöpfe der acht möglichen Sprites.

Bit null entspricht dem An- und Ausschaltknopf für Sprite null, Bit 1 ist für Sprite 1 zuständig und so weiter. Wir wollen Sprite null und eins anschalten, dazu setzen wir die Bits null und eins und erhalten den Wert 3. Wenn wir in die Speicherstelle 53269 (Zeile 250) den Wert 3 einPOKEn, schalten sich beide Sprites an. Eine 2 würde nur den Lastwagen sichtbar machen und die 1 bekanntermaßen das Cabrio.

```
250 POKE 53269,3
```

Der Lastwagen soll von rechts nach links über den Bildschirm fahren, das Cabrio kommt aber von links. Für den Lastwagen müssen wir den Befehl aus Zeile 300 umkehren. In Zeile 300 zählt der C 64 von 0 bis 255, die sich vergrößernden Werte lassen das weiße Auto von links nach rechts fahren.

Beim Lastwagen müssen die Werte für X immer kleiner werden. Das Cabrio fährt für X von 0 bis 255, der Lastwagen soll von 255 bis 0 fahren. Zu diesem Zweck drehen wir den Befehl um, die X-Koordinate des Lastwagens berechnet sich aus 255-X.

Je größer X wird, um so weiter fährt der Brummi nach links. Das müssen wir uns in Ruhe durch den Kopf gehen lassen.

Wo soll der Lastwagen erscheinen? Die Koordinaten von Sprite 1 kommen in die Speicherstellen 53250 (X-Koordinate Sprite 1) und 53251 (Y-Koordinate Sprite 1), genau hinter die von Sprite 0. Die neue Zeile heißt

```
320 POKE 53250,255-X:POKE 53251,100
    :REM KOORDINATEN SPRITE 1
```

Starten wir das Programm. Prima, die beiden Sprites fahren in entgegengesetzter Richtung über den Bildschirm. Jetzt wird klar, warum die Sprites verschiedene Nummern haben.

Das weiße Cabriolet (Sprite Nummer 0) fährt vor dem Laster mit der Nummer 1 her, es überdeckt die Farbe des Lastwagens. Es ist Zeit für einen weiteren Beweis unseres Könnens. Die Farbe von Sprite 1 legen wir in Speicherplatz 53288 fest, genau eine Byte-Position hinter der Farbe von Sprite 1. Den neuen Befehl setzen wir direkt hinter den Farb-Befehl von Sprite 0

```
270 POKE 53288,0:REM FARBE SPRITE 1
```

Damit wir die verschiedenen Speicherstellen nicht durcheinanderwerfen, habe ich in Tabelle 18.2 eine Liste zusammengestellt, in der alle wichtigen Bytes aufgeführt sind.

```
POKE 2040,11          Codezahl für Speicherplatz Sprite 0
POKE 2041,13          Codezahl für Speicherplatz Sprite 1
POKE 53269,X          schaltet die Sprites an
POKE 53287,X          setzt die Farbe für Sprite 0
POKE 53288,X          setzt die Farbe für Sprite 1
POKE 53289,x          setzt die Farbe für Sprite 2
POKE 53890,X          setzt die Farbe für Sprite 3
POKE 53248,X          X-Koordinate für Sprite 0
POKE 53249,X          Y-Koordinate für Sprite 0
POKE 53250,X          X-Koordinate für Sprite 1
POKE 53251,X          Y-Koordinate für Sprite 1
POKE 53252,X          X-Koordinate für Sprite 2
POKE 53253,X          y-Koordinate für Sprite 2
POKE 53253,X          X-Koordinate für Sprite 3
POKE 53254,X          Y-Loordinate für Sprite 3
```

Tabelle 18.2: Alle Speicherstellen der Sprite-Programmierung in Tabellenform.

Wüstenstraße

Leute, unser Grafik-Programm ist fertig. Wie versprochen, fahren ein schwarzer Lastwagen und ein weißes Cabrio über den Bildschirm. Als Krönung der Sache verschönern wir den Bildschirm nach Belieben mit einer Straße und Häusern. Wer will, kann Fabriken oder Kakteen zeichnen, zwischen denen die zwei Fahrzeuge ihre Bahnen ziehen.

Das Prinzip ist ganz einfach: Der PRINT-Befehl zeichnet jede beliebige Landschaft auf den Bildschirm, einige neue PRINT-Zeilen am Anfang des Programms genügen. Vorne auf den Tasten befinden sich die Grafikzeichen, die mit SHIFT- oder COMMODORE-Taste erreichbar sind. Wir verschönern das Programm mit drei Zeilen, die den PRINT-Befehl und die Grafik-Zeichen ausnutzen.

```
210 PRINT "SHIFT/CLR-HOME"
220 PRINT "(7*Cursor runter)
           (40*SHIFT/R)":REM STRASSE
230 PRINT "(4*Cursor rechts)
           (3*Cursor rauf)
           (CBM/Z)(CBM/X)
           (1*Cursor rauf)
           (2*Cursor links)
           (CBM/Q)(CBM/W)
           (1*Cursor rauf)
           (2*Cursor links)
           (SHIFT/N)
           (SHIFT/M)":REM HAUS
```

Die in Klammern stehenden Ausdrücke sind Tastenkombinationen. (7mal Cursor runter) zum Beispiel bedeutet: Drücke siebenmal die Taste für Cursor-runter. Auf dem Bildschirm erscheinen sieben inverse Q. CBM bedeutet »Commodore«-Taste. Ihr findet sie ganz links unten auf der Tastatur.

Probiert das Ganze mal aus. Die Autos fahren auf einer Straße und daneben steht ein Haus aus Grafikzeichen. Ihr könnt das Haus an jede Stelle des Bildschirms führen, ein Verändern der Cursor-Befehle genügt.

Listing 18.1 zeigt das ganze Programm auf einen Blick. Bevor Ihr es neu abtippt oder von der Buch-Diskette einladet, schaltet den C 64 kurz aus. Auf diese Weise wird der Grafik-Speicher gelöscht, sonst kann es zu Überlagerungen zwischen alten und neuen Speichereingaben kommen. Das ist nicht schlimm, sieht aber unschön aus.

Hey Freunde, wir haben unser eigenes Grafik-Programm geschrieben. Wir können das Auto-Programm ohne Probleme erweitern, der C 64 hat jede Menge verborgener Fähigkeiten.

Zum Schluß schaue ich mir noch einmal die Tabelle mit den für die Sprite-Programmierung wichtigen Speicherstellen an. Zufrieden lehne ich mich zurück und starte mein Sprite-Programm.

```
210 PRINT"{CLR}":REM BILDSCHIRM LOESCHEN   <191>
220 PRINT"{7DOWN}RRRRRRRRRRRRRRRRRRRRRRRRR
    RRRRRRRRRRRRRRR":REM STRASSE           <103>
230 PRINT"{4RIGHT,3UP}ZX{UP,2LEFT}QW{UP,2L
    EFT}NM":REM HAUS                       <012>
240 POKE 2040,11:POKE 2041,13:REM SPRITE 0
     UND 1                                 <167>
250 POKE 53269,3:REM BEIDE SPRITE AN       <244>
260 POKE 53287,1:REM FARBE SPRITE 0        <167>
270 POKE 53288,0:REM FARBE SPRITE 1        <176>
280 FOR N=0 TO 26:READ Q:POKE 734+N,Q:NEXT <113>
290 FOR T=0 TO 35:READ P:POKE 853+T,P:NEXT <107>
300 FOR X= 0 TO 255                        <008>
310 POKE 53248,X:POKE 53249,100:REM KOORDI
    NATEN SPRITE 0                         <124>
320 POKE 53250,255-X:POKE 53251,100:REM KO
    ORDINATEN SPRITE 1                     <064>
330 POKE 53277,2:REM X-ACHSE GROESSER      <174>
340 POKE 53271,0:REM Y-ACHSE GROESSER      <193>
350 NEXT X                                 <044>
360 GOTO 300                               <050>
370 DATA 0,0,0,0,3,0,0,0,128,0,0,64        <240>
380 DATA 127,255,254,127,255,252,120,127,1
    4,123,127,110                          <044>
390 DATA 3,0,96                            <212>
400 DATA 0,0,56,0,0,68,7,224,130,4,33,128  <248>
410 DATA 4,33,0,63,225,0,63,225,28,127,225
    ,28                                    <002>
420 DATA 127,255,254,97,254,2,109,254,218,
    12,0,218                               <092>
```


Listing 18.1: Unser Sprite-Programm.

Das haben wir gelernt

Sprites: Oft auch »Shapes« genannt, sind grafische Elemente, die selbst definiert und über den Bildschirm bewegt werden können. Für die Programmierung von Spielen kann man die verschiedensten Figuren (Raumschiffe, Monster, Clowns etc.) entwerfen. Mit einfachen POKEs können sie bewegt, vergrößert oder verkleinert werden.

Kapitel 19

Wir lassen Register erklingen

Ein kleines Musik-Programm entspringt unseren begabten Fingern und Köpfen. Die Grundlagen haben wir in Kapitel 10 kennengelernt, jetzt stellen wir sie auf eigene Beine. Die Sache ist nicht schwer, Schritt für Schritt entsteht ein Ton. Bald treten wir im Fernsehen auf: die »Commodore Singers«.

Fassen wir unser Wissen kurz zusammen: Der C 64 besitzt einen Baustein zur Musik-Programmierung. Er nennt sich SID (Sound Interface Device) und beginnt ab Speicherstelle 54272. Mit Hilfe der Speicheradressen des SID (Register genannt) kann der C 64 bis zu drei Stimmen gleichzeitig ertönen lassen. Jede Stimme hat reservierte Register, in denen die besonderen Merkmale der produzierten Töne niedergeschrieben sind. Der eine Ton ist hoch und quäkend, ein anderer klingt wie ein startendes Raumschiff. Welche Töne und Geräusche der Bit-Onkel vor uns auf dem Tisch herstellen kann, hat er in Kapitel 10 bewiesen. Jetzt beweisen wir ihm, was wir alles können.

Wenn die Töne laufen lernen

Als erstes wollen wir dem Computer einen einzigen Ton entlocken. Vorher rücken wir den Geheimnissen der Soundprogrammierung auf den Pelz. Pelzen wir los. Der C 64 kann drei Stimmen produzieren. Die besonderen Tonmerkmale werden in je sieben Registern abgelegt. Wir werden uns im wesentlichen mit der ersten Stimme befassen. Die Register für diese Stimme sind

```
54272=S+0 (Basisadresse des SID)
54273=S+1
54274=S+2
54275=S+3
54275=S+4
54272=S+5
54277=S+6
```

Bis auf die Register S+2 und S+3 benötigen wir für unser Tonprogramm alle Speicherstellen der ersten Stimme. Was kommt in die Register hinein? Bild 19.1 zeigt uns eine Auflistung der wichtigsten Speicheradressen.

Die Register des SID

Basis-Adresse S=54272

Stimme 1	Stimme 2	Stimme 3	7	6	5	4	3	2	1	0
S+0	S+7	S+14	Low-Byte							
S+1	S+8	S+15	High-Byte							
S+2	S+9	S+16	Pulsweite-low							
S+3	S+10	S+17					Pulsweite-high			
S+4	S+11	S+18					Test	Ring	Sync	Gate
S+5	S+12	S+19	Attack (x16)				Decay			
S+6	S+13	S+20	Sustain (X16)				Release			
		S+24						Lautstärke		

Bild 19.1: Die Register des SID beherbergen die Ton-Informationen.

Auf der linken Seite sind die Register der drei Stimmen aufgelistet. Jede Stimme hat die gleichen Register, nur die Nummer ist anders. Deswegen ist es im Moment egal, ob wir uns mit der ersten oder der dritten Stimme beschäftigen. Register 24 fällt etwas aus dem Rahmen: Die Lautstärke wird für alle drei Stimmen gleichzeitig festgelegt.

Oben an der Tabelle sind die einzelnen Bits dargestellt. Der Klang wird durch An- oder Ausschalten verschiedener Bits bestimmt. Die Bit-Nummern gehen von Bit 0 bis Bit 7. Register S+0 nimmt das Lo-Byte auf, S+1 das Hi-Byte. Wie wir wissen, werden Tonhöhen in Zahlen angegeben. In Kapitel 9 hatten wir zum Beispiel die Tonhöhe 40000 (das sehr hoch und fein klingende Glöckchen). Diese Zahl ist für den Computer die Grundlage für den zu spielenden Ton. Sie muß in die entsprechenden Register eingePOKEt werden. Ein Register kann nur Zahlen bis 255 aufnehmen! Selbst bei Anschalten aller Bits ruft jede Zahl, die größer als 255 ist, eine »ILLEGAL QUANTITY«-Fehlermeldung hervor. Die Tonhöhe muß in Werte zerlegt werden, die kleiner als 256 sind: das High- und das Low-Byte.

Leute, keine Sorge, die Zerlegungs-Formel interessiert uns nicht die Bohne. Alle für unser Musik-Programm benötigten Werte kennen wir schon. Wie wir in Bild 19.1 sehen, kommt das Low-Byte in Register S + 0 und das High-Byte in Register S + 1. In

einem vollständigen Musik-Programm liest das schlaue Kerlchen vor uns die beiden Werte und setzt aus ihnen selbständig den Ton zusammen. So einfach ist das!

Klaviergehämmer

Die ersten beiden Register sind geklärt. Wenden wir uns den Speicherstellen S+5 und S+6 zu. In Bild 19.1 sehen wir in diesen Registern irgendwelche turnenden Raupen und daneben unverständliche Worte: Attack, Decay, Sustain, Release. Der Sinn dieser Begriffe lädt uns zu einem kleinen Ausflug in die Feinheiten des Klavierspielens ein. In Kapitel 10 haben wir in diesem Zusammenhang von »Tasten-Anschlag« gesprochen, diese Umschreibung sehen wir uns genauer an.

Nehmen wir folgenden Fall an: Vor uns steht ein wunderschönes schwarzes Klavier mit ebensoschöner Tastatur. Wir wollen ausprobieren, wie sich ein einziger Ton des Klaviers durch verschieden starkes Drücken der Taste verändern läßt. Im ersten Versuch drücken wir kurz und kräftig die Taste und lassen sie danach wieder los: Der Ton entsteht und fast im gleichen Moment wird er wieder leiser. Ein zweiter Versuch: Die gleiche Taste wird kurz und hart gedrückt, aber nicht sofort wieder losgelassen. Der Ton entsteht wie beim ersten Versuch, bleibt durch die gedrückte Taste viel länger hörbar. Beim dritten Mal schlagen wir volle Kanne auf die Taste: hart und laut entsteht der Ton. In einem letzten Versuch drücken wir die Taste so sanft und leicht wie möglich: Ein kaum hörbarer Ton erklingt.

Dieser kleine Test zeigt uns: Ton ist nicht gleich Ton. Eine Taste kann sehr verschiedene Klänge erzeugen. Je nach Druck der Taste entstehen lange oder kurze, harte oder weiche Klänge. Die Raupe in den Registern S+5 und S+6 stellt den Klangverlauf eines Tons dar. Wie beim Klavier, können wir mit dem C 64 den Klang eines Tons festlegen.

Aus dem Leben einer Ton-Raupe

Bild 19.2 zeigt uns eine Klang-Raupe in Großaufnahme. Es erklärt die vier wichtigen Phasen im Leben eines Tons. Am Anfang wird die Taste gedrückt. Kurz nach dem Anschlag hat der Ton seinen lautesten Moment, von jetzt ab wird er immer leiser. Zwischen dem Anschlag und dem lautesten Moment vergeht eine bestimmte Zeitspanne. Je schneller der höchste Punkt erreicht wird, desto härter war der Anschlag der Taste. Bei einem weichen Anschlag kann sich der Ton »entfalten« und braucht zum Erreichen des Höhepunkts längere Zeit. Der Moment zwischen Drücken der Taste und dem lautesten Moment des Tons wird mit dem Begriff »Attack« umschrieben (deutsch: »Anschlag, Beginn«) und ist in Bild 19.2 unter der Raupe mit »A« abgekürzt. Nach Attack kommt eine Phase, in der der Ton an Stärke verliert, bevor er sich auf einer bestimmten Lautstärke stabilisiert. Diese Phase heißt »Decay« (deutsch: »schwächer werdend«). Decay geht in »Sustain« über (deutsch: »Halten«). Dieser Abschnitt kann

am ehesten mit dem Gedrückt-Halten der Klaviertaste verglichen werden: Je länger die Klaviertaste gedrückt ist, desto länger wird die Sustain-Phase. Der letzte Abschnitt trägt den Namen »Release« (deutsch: »Fallen«). Der Ton wird immer schwächer und verschwindet schließlich ganz.

Bild 19.2: Die ADSR-Kurve bestimmt den Verlauf des Tones.

Der Ton, den wir produzieren wollen, hängt stark von den gewählten Werten für die Phasen ab. Jeder Abschnitt kann durch einen Zahlenwert frei definiert werden. Die Zahlen für Attack und Decay kommen in Register S+5, die von Sustain und Release in Register S+6. Der entsprechende Abschnitt ist durch eine schwarze Verdickung der turnenden Raupen zu sehen. Für alle vier Phasen können Werte von 0 bis 15 eingegeben werden. Die Einstellungen für Attack und Sustain müssen dabei mit 16 multipliziert werden. Hier ein Beispiel

```
POKE 54272+5, 1*16+9:REM ATTACK/DECAY
```

Der Wert für Attack ist 1, der für Decay 9.

```
POKE 54272+6, 4*16+4
:REM SUSTAIN RELEASE
```

Sustain hat die Eingabe 4, Release liegt ebenfalls bei 4. Mit Bild 19.2 verglichen, ergibt das einen Ton, der schnell anschwillt (kleiner Wert für Attack = harter Anschlag), etwas langsamer leiser wird (Decay), sich bei einem niedrigen Niveau hält (Sustain) und dann recht schnell verklingt (Release).

Je größer die Werte für Attack, Decay und Release sind, desto breiter wird der mit »A«, »D« und »R« gekennzeichnete Bereich in Bild 19.2. Für »A« bedeutet ein hoher Wert (zum Beispiel 14x16) einen Ton, der langsam seinen Höhepunkt erreicht.

Je kleiner die Eingabe, desto enger rücken die Bereiche zusammen. Sustain verhält sich etwas anders. Die Angabe von 0 bis 15 legt hier die »Höhe« des Haltepegels fest. In Bild 19.2 ist diese Tatsache durch die senkrechten Pfeile und »S« gekennzeichnet. Wenn wir für Sustain 15x16 eingeben, so rückt der Haltepegel ganz nach oben. In diesem Fall ist der höchste Punkt von Attack gleich Sustain, es gibt kein Decay mehr, weil der Ton zwischen Attack und Sustain nicht abnimmt. Wenn Sustain gleich 0x16 gesetzt wird, so entfällt der Wert für Release, da der Ton bei Decay bis auf den Wert 0 zurückfällt.

Ein Ton ist ein kniffliges Gebilde, aber wir sind kniffliger! Als halbe Tonfachmänner machen wir uns weiter auf den Weg, ein gutes Stück haben wir bereits zurückgelegt.

Ein Ton sägt sich durchs Trommelfell

Die beiden POKE-Zeilen da oben behalten wir im Kopf, sie sind ein Teil unseres ersten Programms. Im ersten Programm wollen wir einen einzigen Ton erzeugen, dafür benötigt der C 64 neben dem Hi-Byte, Lo-Byte und den Werten für A, D, S, R weitere Informationen. Bild 19.1 zeigt uns bei Register S+4 bisher unbekannte Dinge. Es ist in einzelne Bytes unterteilt. Bisher haben wir die Tonhöhe und den Tonverlauf festgelegt. In Register S+4 geben wir den Startbefehl des Tons und seine »Wellenform« an. Sie bestimmt die Klangart des Tons. Er kann wie eine schlechtgelaunte Ente quäken, wie ein Schuß oder zerreißender Stoff klingen, aber auch leicht und empfindsam singen. Die Klangart wird mit den vier Zeichen unter den Bits 4 bis 7 festgelegt. Das Setzen des dazugehörigen Bits läßt den gewünschten Klang entstehen. Bit 7 enthält die Wellenform »Rauschen«: Der erzeugte Ton klingt wie ein Schuß, eine startende Rakete oder wie das Rauschen des Radios, je nach Einstellung von A, D, S, R. In Bit 5 befindet sich die sogenannte »Sägezahn«-Wellenform, mit der ein Ton hart und quäkend klingt.

Wenn wir Bit 5 setzen, erhalten wir einen quäkenden harten Ton (Bit 6 nennt sich »Rechteck«-Wellenform, interessiert uns im Moment aber nicht. Zur Benutzung von »Rechteck« müssen die Register S+2 und S+3 zusätzlich gesetzt werden: S+2 nimmt Zahlen bis 255 auf, S+3 Werte bis 15). Für unseren ersten Ton entscheiden wir uns für die »Dreieck«-Wellenform, mit der ein weicher, feiner, oboenhafter Klang erzeugt wird. Die Dreieck-Wellenform befindet sich in Bit 4. Für unseren ersten Ton setzen wir dieses Bit: Bit 4 bedeutet 2^4=16! Das Bit, das den Ton anschaltet (und hinterher wieder ausschaltet) befindet sich in Nummer 0 (Bit 0: 2^0=1).

EinPOKEn von 17 (16+1) in Register S+4 legt eine Dreieck-Wellenform fest und schaltet den Ton ein. Gegen Ende des Programms muß das Einschalt-Bit wieder zurückgesetzt werden, das geschieht durch EinPOKEn von 16 (16+0).

Kitzeln im Ohr

Die Post geht ab, alle wichtigen Dinge für das Ton-Programm sind geklärt! Ein Programm entsteht. Gehen wir die einzelnen Schritte durch:

1. Lautstärke: Register S+24
2. Attack und Decay: Register S+5
3. Sustain und Release: Register S+6
4. Hi- und Lo-Byte einPOKEn: Register S+1; Register S+0
5. Wellenform, Gate on (Ton einschalten): Register S+4
6. Eine FOR-NEXT-Schleife einbauen, damit der Ton sich eine Weile hält und wir ihn richtig hören können
7. Gate (Bit 0) zurücksetzen: Register S+4

Wenn wir diese Punkte in Befehlszeilen übertragen, erhalten wir ein vollständiges Programm, das einen Ton produziert. Listing 19.1 zeigt Euch die funktionsfähige Version (auch auf Diskette!). In Zeile 60 befindet sich die FOR-NEXT-Schleife: Solange der C 64 von 0 bis 500 zählt, können wir den Ton hören. Befehl 70 setzt Bit 0 zurück und schaltet den Klang aus.

```
1 REM HENNING SPIELT EINEN TON                <155>
5 S=54272:REM BASISADRESSE SID                <252>
10 POKE S+24,15:REM LAUTSTAERKE               <136>
20 POKE S+5,1*16+9:REM ATTACK DECAY           <211>
30 POKE S+6,4*16+4:REM SUSTAIN RELEASE        <174>
40 POKE S+1,29:POKE S+0,69:REM HI/LO-BYTE     <251>
50 POKE S+4,17 :REM DREIECK GATE ON           <176>
60 FOR R=1 TO 500:NEXT:REM TONDAUER           <231>
70 POKE S+4,16:REM GATE ZURUECK               <100>
```


Listing 19.1: Musik-Programm.

Anhand dieses Programms probieren wir die bisher theoretischen Kenntnisse aus. Zuerst wählen wir eine andere Wellenform, statt Dreieck nehmen wir Rauschen. Das Setzen von Bit 7 ergibt den Wert 128 ($2^7=128$) plus 1 von Bit 0 (Gate on): In Zeile 50 wird 129 eingePOKEt, in Befehl 70 128. Als Ergebnis erhalten wir den Klang eines harten Schusses. Das ist kein Wunder, denn der Wert für Attack ist 1, der Ton entsteht sehr schnell und hart. Wenn dieser Wert zum Beispiel auf 15 geändert wird, klingt der Ton wie Sand, den man aus einem Plastikeimer schüttet: Die Attack-Phase ist viel länger! Ein weiterer netter Effekt stellt sich durch Einsetzen von 15 für Release ein. Der

Ton wird ganz langsam leiser. Wer will, kann jetzt die Wellenform in »Sägezahn« umändern: Zeile 50 nimmt 65 auf, Befehl 70 64. Viel Spaß dabei!

Das haben wir gelernt

ADSR-Kurve: Der Verlauf eines Tones kann beim C 64 mit Hilfe der vier Werte für A, D, S und R genau festgelegt werden. A steht für Attack, D für Decay, S für Sustain und R für Release. Die Werte von Attack und Sustain werden mit 16 multipliziert.

Wellenform: Neben dem Tonverlauf kann die Art des Tons bestimmt werden. Zu diesem Zweck stehen bestimmte Zahlenwerte zur Verfügung, die dem Ton einen charakteristischen Klang geben. Es gibt Rauschen, Rechteck, Sägezahn und Dreieck.

Stichwortverzeichnis

64'er Großer Einsteiger-Kurs

Frisch ausgepackt steht der C64 vor Ihnen. Voller Ehrgeiz starten Sie Ihre ersten Versuche am eigenen Computer. Doch der schnell erhoffte Erfolg bleibt aus. Solche Erfahrungen machte auch Henning Withöft, aber er hat durchgehalten, bis sich zu den ersten Programmier-Erfolgen auch das Verständnis für die Computer-Technik einstellte. Seine lehrreichen Erfahrungen, Erfolge und auch Fehlschläge dokumentierte er für die große Schar der C64-Einsteiger.

So entstand ein Buch, das den Computer-Neuling Schritt für Schritt durch die »neue Welt« führt. Angefangen vom Auspacken und Anschließen des C64, über Basic-Programmierung bis zu PEEK- und POKE-Befehlen oder der Sprite-Programmierung wird alles für den Einsteiger Wissenswerte über den C64 behandelt – ein Buch vom Einsteiger für den Einsteiger, anschaulich mit erklärenden Diagrammen und Beispiel-Listings, geschrieben in einer lockeren, dem Leser leichtverständlichen Sprache.

Ohne Zweifel haben die Autoren die Einsteiger-Serie »Henning packt aus« aus dem 64'er-Magazin zu einem vollwertigen Kurs in Buchform weiterentwickelt.

Auf der beiliegenden Diskette finden Sie alle größeren Beispiel-Programme und ein Spieledemo von Giana Sisters – ein kleiner Einblick in das, was alles mit dem C64 möglich ist.

Hardware-Anforderungen:

C64 oder C128 im 64er-Modus mit einer Floppy (1541, 1570, 1571)

HENNING WITHÖFT, geboren 1966, kam erst nach dem Abitur in späten, aber gründlichen Kontakt mit dem C64. Seitdem ist es sein erklärtes Ziel, Einsteigern die ersten Schritte im neuen Hobby Computer zu erleichtern. Withöft ist heute als Journalist tätig.

ANDREW DRAHEIM, geboren 1965 in Düsseldorf, erkannte nach dem Abitur seine journalistische Ader und wurde freier Mitarbeiter bei der Aachener Volkszeitung, bis er 1987 zum 64'er-Magazin wechselte, dort ist er seitdem für den Bereich »Einsteiger« und Leserforum zuständig.

ISBN 3-89090-668-0

02990

4 001057 906686

DM 29,90 sFr 27,60 öS 233,–

Markt&Technik